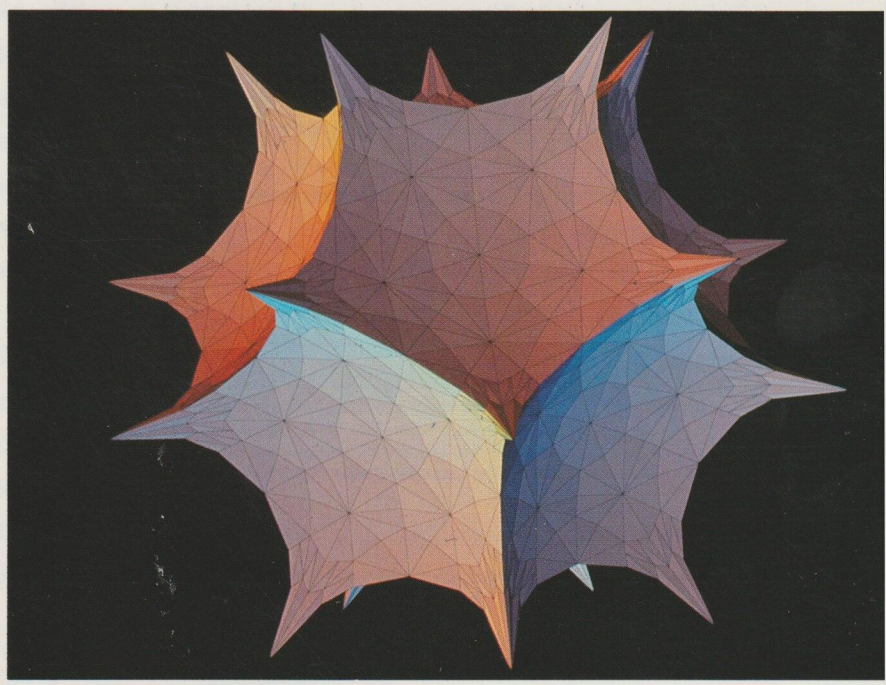
Langages & systèmes

MARS 1992 - N° 6 - SUPPLEMENT DÉVELOPPEURS DU N° 79 D'INFOPC - ISSN 0951- 6402

TIGA

La norme graphique de Texas Instruments est en passe de devenir un standard en matière d'affichage haute résolution



LANGAGES ORIENTÉS OBJET

Borland s'implique

APPRENDRE WINDOWS

Etude de l'interface graphique GDI de Windows

ET AUSSI...

- LA DSP : ALTERNATIVE AUX RISC ?
- LES PROFILERS : OPTIMISER SON CODE EXÉCUTABLE
- CLIPPER : LE COMPILATEUR NANTUCKET GAGNE EN PUISSANCE

INFO'PC
LE MAGAZINE DE LA MICRO-INFORMATIQUE PROFESSIONNELLE

NE PEUT ÊTRE VENDU SÉPARÉMENT

Nouveau Microsoft vos micros vont un cerveau particu

Plus qu'un serveur, un véritable cerveau !

SQL Server version 4.2 offre aux réseaux locaux les fonctionnalités des moteurs de bases de données relationnelles qui n'existaient jusqu'alors que sur mini-ordinateurs ou sites centraux.

Depuis toute station Windows, l'administrateur peut superviser les serveurs SQL Server du réseau, gérer les accès des utilisateurs, traiter des requêtes ou planifier les sauvegardes des données.

Les caractéristiques avancées de SQL Server 4.2 en font, dans l'univers de la micro-informatique, le seul serveur de bases de données relationnelles intelligent.

L'intelligence de SQL Server permet d'incorporer au sein même de la base de données, les règles d'intégrité des données et les contraintes propres à l'entreprise, plutôt que d'avoir à programmer chaque application frontale pour qu'elle respecte ces différentes règles.

L'avantage de cette approche est que toutes les règles sont centralisées et que le point de contrôle unique pour la cohérence des données est situé sur le serveur. Quand les règles évoluent, il n'est donc plus nécessaire de modifier toutes les applications frontales, il suffit de répercuter ces changements sur le serveur. Le développement et la maintenance des applications s'en trouvent simplifiés et la sécurité des données sera effectivement assurée quelle que soit l'application frontale utilisée.

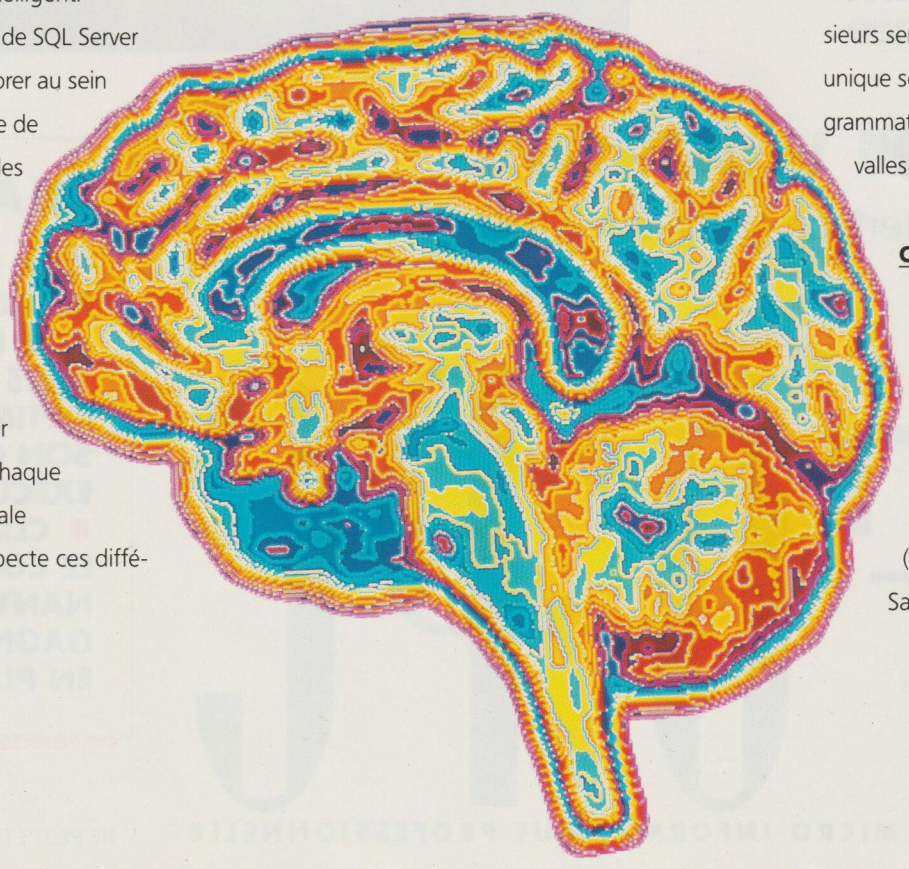
SQL Server 4.2 est conçu pour opérer 24 heures sur 24 dans des applications critiques, et permet de faire des sauvegardes dynamiques sur bande ou sur disque alors que les utilisateurs continuent de travailler.

Un outil d'administration complet sous Windows.

SQL Server 4.2 propose un nouvel outil d'administration qui permet de gérer les utilisateurs, les permissions d'accès, la sauvegarde des données et de l'environnement. Il autorise également des connexions simultanées à plusieurs serveurs depuis une station unique sous Windows, ainsi que la programmation de sauvegardes à intervalles réguliers.

Le nec plus ultra de la connectivité.

SQL Server 4.2 permet de s'interfacer avec des bases de données relationnelles ou non relationnelles situées sur minis ou grands systèmes grâce à ses services ODS (Open Data Services). Sans qu'il soit nécessaire de



ROUX SEGUELA CAYZAC & GOUDARD Photo Cosmos

SQL Server 4.2 : être en liaison avec lièrement brillant.

modifier une application frontale accédant à SQL Server, il est possible de permettre un accès transparent à des bases de données externes ou encore de déclencher l'exécution d'un programme prédéfini.

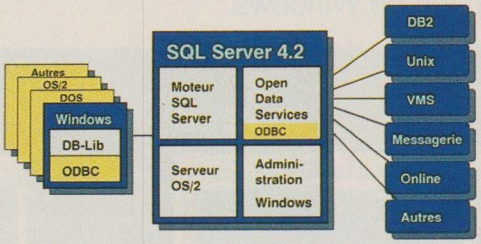
En cas de modification du niveau d'un stock dans la base de données, par exemple, SQL Server déclenchera automatiquement l'envoi d'un message à la personne responsable des achats via la messagerie électronique de l'entreprise, si le niveau du stock tombe en-dessous d'un seuil prédéterminé.

Outre Microsoft LAN Manager, SQL Server est compatible avec les plus grands systèmes d'exploitation réseaux du marché. Le protocole de communication de SQL Server a en effet été adopté par les principaux éditeurs de produits réseaux : Novell Netware, IBM LAN Server...

Une ouverture en toute sécurité.

SQL Server 4.2 est ouvert à un très grand nombre d'applications frontales, permettant ainsi aux utilisateurs de conserver l'environnement qui leur est familier.

Les coûts de formation en sont sensiblement réduits et l'investissement initial de l'entreprise n'est pas remis en cause.



A travers les services ODS de SQL Server 4.2, les stations clientes accèdent de façon transparente à des applications existantes telles que des bases de données relationnelles ou non, des services de messagerie...

A ce jour, plus de 90 applications ont été répertoriées autour de SQL Server. Parmi les applications principales on trouve notamment : Microsoft Excel, Visual Basic, SQL, Windows, dBase, Paradox, Lotus 1-2-3, Superbase, Omnis 5, Objectview...

Cette ouverture exceptionnelle est de plus complètement sécurisée étant donné que SQL Server gère lui-même l'intégrité des données plutôt que de laisser ce soin aux différentes applications frontales utilisées.

La plate-forme de production micro.

SQL Server 4.2, grâce à ses caractéristiques de sécurité, d'ouverture et de performances, représente une réelle plate-forme de production dans le domaine de la micro-informatique. Cette plate-forme présente des performances similaires à celles des architectures multi-utilisateurs classiques et ce pour un rapport prix/performance particulièrement avantageux du fait de l'utilisation de micro-ordinateurs standards du marché.

Pour toutes informations complémentaires ou recevoir la liste des Revendeurs Spécialistes Réseaux Microsoft :
- appelez le service Clients au (1) 69 86 46 46
- ou utilisez le Minitel : 3616 code Microsoft.

Microsoft, 91957 Les Ulis Cedex
et à Infomart La Défense
Stand n° 276 niveau 2.

Service lecteurs n° 1 (dernière page)

Toutes les marques citées sont des marques déposées.

Microsoft®

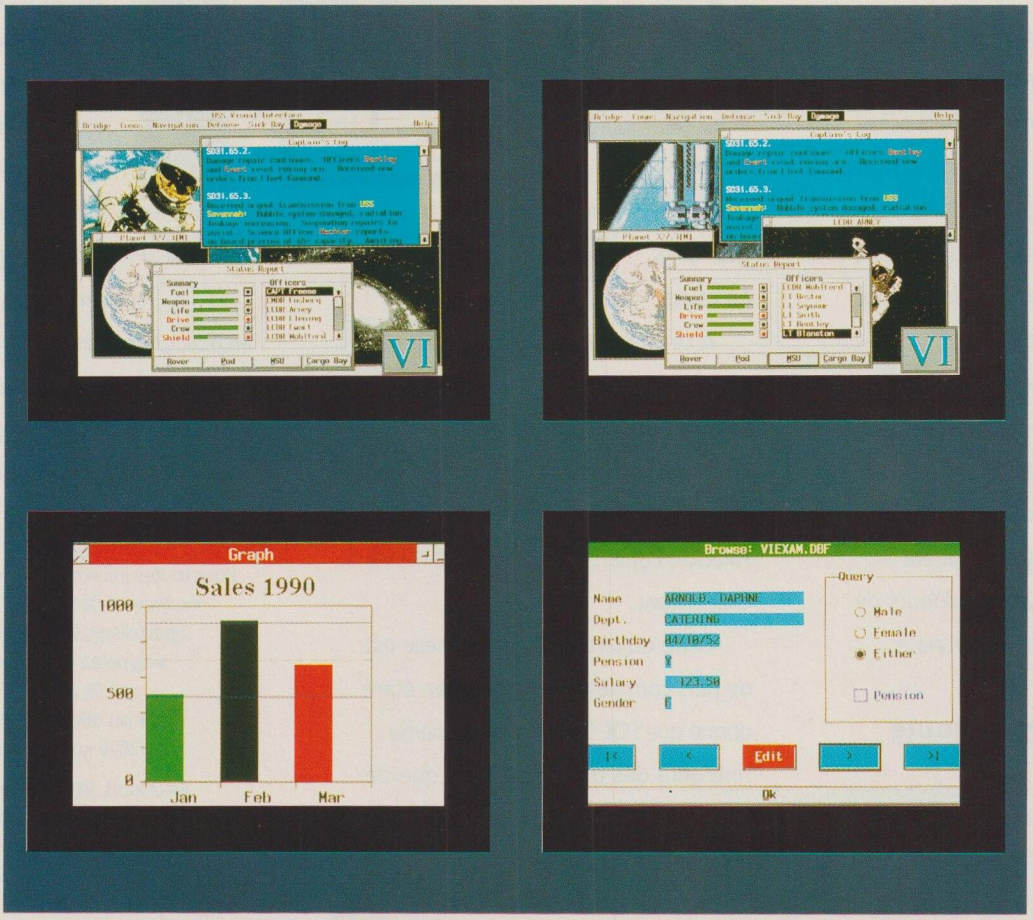
LES LOGICIELS QUI DONNENT DES AILES.

Visual Interface pour Nantucket 5

VI (Visual Interface) pour Nantucket 5 est une interface utilisateur graphique à la Windows qui vous offre pour la première fois les 3 fonctionnalités indispensables pour vos applications :

- ☛ Des fenêtres haute résolution avec les contrôles interactifs
- ☛ Un gestionnaire d'évènement intégré
- ☛ Des fenêtres, des objets graphiques, et des icônes qui s'intègrent parfaitement à Clipper 5

Les applicatifs peuvent être exécutés à partir du DOS 3 ou plus, ou lancés à partir de Windows.



Nantucket 5 et VI : les environnements de développement en mode graphique

Je désire recevoir la documentation concernant le produit Visual Interface pour Clipper

M, Mme, Mlle .....Prénom .....

Société .....Fonction .....Adresse.....

.....

Code Postal.....Ville.....Téléphone.....

PC Technologie - 4/4 bis, rue Bourgeoise - 92800 PUTEAUX
Tél : (1) 42 04 00 88 - Fax : (1) 42 04 69 80

PC TECHNOLOGIE

BITS PER SECOND

Service lecteurs n° 4 (dernière page)

Sommaire

- 7 -

Re-sources

Une sélection des nouveaux outils de développement.

- 16 -

Tiga

Des processeurs graphiques très performants, pilotés par des primitives accessibles par le biais d'une bibliothèque appelée «Tiga»

- 24 -

Apprendre Windows

Présentation de l'interface graphique de Windows. Celle-ci peut fournir des graphiques indépendants du périphérique.

- 42 -

Driver sous DOS

L'écriture d'un driver est moins complexe qu'il n'y paraît. Comme exemple, un driver pilotant le port série RS232-C

- 56 -

C++ ou Pascal

Borland a basculé dans le camp des langages orientés objet et propose deux compilateurs ayant une finalité bien distincte.

- 62 -

Les Profilers : Microsoft contre Borland

Les Profilers fournissent les informations nécessaires à l'optimisation du code d'un programme exécutable.

- 66 -

Clipper 5.0

Le compilateur Clipper dans sa version 5.0 apporte des extensions qui étendent encore son champ d'application.

- 70 -

DSP contre RISC

Les DSP sortent de leur domaine réservé, pour concurrencer les processeurs RISC dans les développements spécifiques.

- 76 -

Concours GOLD

Participez dès maintenant à la seconde mouture du Grand Open Logiciel et Développement organisé par *Langages & Systèmes*.

- 78 -

Enquête lecteur

Langages & Systèmes se met à l'écoute de ses lecteurs.

Chronique d'échecs annoncés

La courte histoire de la micro-informatique est parsemée d'annonces fracassantes suivies d'échecs tout aussi retentissants. Ainsi, l'association Microsoft-IBM pour préparer l'après-MS-DOS. Ces deux sociétés ont codéveloppé un système d'exploitation 16 bits qui s'est révélé un «flop» commercial : OS/2. Aujourd'hui, IBM tente de le relancer seul, après l'avoir complètement réécrit. L'OS/2 nouvelle cuvée présente des avantages bien réels face au couple MS-DOS/Windows 3.0. Mais il n'existe encore qu'à l'état de bêta version tandis que Windows 3.0 connaît de son côté un succès inespéré (près de 10 millions d'exemplaires vendus). Faire des pronostics sur l'avenir d'OS/2 2.0 ne relève donc pas des talents d'un cartomancien : l'exemple de Digital Research est assez édifiant.

Un autre groupement se trouve en ce moment dans des turbulences : ACE. L'Advanced Computing Environment, créé l'an dernier, vise à «établir le standard de la décennie 90». L'année 1992 à peine entamée, plusieurs constructeurs commencent déjà à se démarquer ACE. L'une des raisons de cette défection provient de la fragile situation financière de la plupart des principaux partenaires, notamment Mips. Or cette société est à l'origine du processeur RISC, au cœur de la machine ACE. Celle-ci a pourtant de quoi séduire. Son processeur R4000 PC cadencé à 50 MHz (bientôt à 67 MHz) délivre une puissance d'environ 40 Specmarks, de loin supérieure à celle obtenue avec le 486DX d'Intel.

Cependant, la performance n'est pas synonyme de pérennité. Le sort réservé au Sirius S1 dans la première moitié des années 80 le prouve bien : cette machine surclassait en tout point le PC, mais ce dernier s'est imposé comme le standard du marché. Faut-il conc lure qu'il vaut mieux graviter autour du noyau Intel-Microsoft? Ils représentent en tout cas une référence stable en ces temps troublés où même les plus grands ne sont plus épargnés.

Yves Guittard

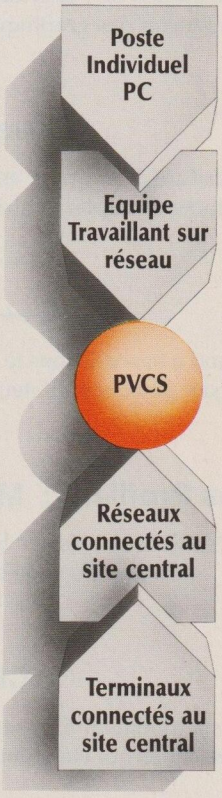
Développer sur PC ou sur réseau local,

c'est facile, c'est pas cher; ça peut rapporter gros,

mais c'est risqué...

parce que les pertes de fichiers, les régressions, les modifications incontrôlées, la difficulté de savoir qui a fait quoi, quand, comment et pourquoi . . . font partie de la vie de tous les jours!

. . . sauf si vous utilisez PVCS d'INTERSOLV.



Standard de fait et leader incontesté de la gestion de versions et de configurations en environnement distribué avec plus de 90.000 utilisateurs, PVCS contrôle tout type d'objets: code source, exécutable, documentation, graphique, etc . . . En outre, il vous garantit de ne jamais perdre ni dupliquer un de ces objets. Enfin, PVCS organise les processus de création ou de modification de programmes grâce à des contrôles appropriés tout au long du cycle de développement.

Conçu pour des réseaux locaux ou des postes individuels, PVCS s'exécute sous MS-DOS, OS/2, UNIX ou AIX.

PVCS est un outil ouvert qui communique avec les produits homologues sur site central; de plus, il s'intègre avec Cobol Workbench de Micro Focus et Programmers Workbench ou c de Microsoft.

PVCS est le seul outil de gestion de versions et de configurations en environnement distribué reconnu par IBM dans l'AD/Cycle.

Faites-en l'expérience vous-même et contactez Patricia Fauconnier au (1) 30 81 30 82.

INTERSOLV

LE SPECIALISTE EN GENIE LOGICIEL QUE VOUS ATTENDIEZ

Recevez notre documentation ou assistez à un séminaire en retournant le coupon à:
INTERSOLV SA, 44 rue Pierre Curie, 78370 Plaisir

Nom _____ SOCIETE _____

Adresse _____

_____ Tél _____ Poste _____

☐ Souhaite une documentation

☐ Souhaite participer à un séminaire de Gestion de Versions et de Configurations

INTERNATIONAL ALLIANCE IBM AD/Cycle™

Service lecteurs n° 5 (dernière page)

Re-sources

Think toujours

Le Think Pascal est un environnement complet de programmation orientée objet sur Macintosh.

Rapide et puissant, ce produit offre, en plus du compilateur, un large éventail d'outils et s'adresse aussi bien aux novices qu'aux professionnels.

Le compilateur génère un code exécutable de grande qualité en un minimum de temps, car les parties inutiles du programme sont éliminées, et des liens s'établissent simultanément.

Il permet de créer des applications de type accessoire de bureau, pilote de périphérique et toute autre ressource incluant le XCMDs d'HyperCard. L'éditeur assure une gestion conviviale et multifenêtrée, où le volume des fichiers peut dépasser 32 ko.

En ce qui concerne la POO, le Think Pascal inclut une importante bibliothèque de classes, des supports pour MacApp, pour les directives de segmentation, la Toolbox, les routines du système et MPW Projector, un browser et un débogueur.

Le browser de classes, totalement intégré, permet de visualiser l'ensemble des classes d'une application, avec leurs interactions. L'examen ou l'édition, la recherche et la définition d'une classe ou d'une méthode perdent leur caractère de tâches ingrates.

Le débogueur gère graphiquement le choix des points d'arrêt, la lecture et la modification des variables durant l'exécution, même pour les objets et les structures du type tableau ou enregistrement. Le gestionnaire de projet, après avoir répertorié tous les composants d'un programme, ne recompilera que les fichiers modifiés.

Think Pascal 3.0

• Plate-forme de développement orientée objet • Système 5 ou ultérieur • 1 à 4 Mo de RAM • Compilation, épuration et liaison rapide • Editeur multifenêtre et de fichiers • Débogueur graphique et Browser de classes • Pilote de périphérique, ressources • Support Toolbox, segmentation mémoire, MPW Projector, MacApp et routines OS • Bibliothèque objets •

Think encore

Le Think C est lui aussi très intéressant pour les professionnels, et même pour les amateurs, du développement sur Macintosh, car la programmation orientée objet, combinée à la puissance et aux possibilités du langage C, leur offre une plus grande liberté à un haut niveau de performances.

Comme le Think Pascal, ce produit comprend un compilateur et un linker, aussi intelligents que rapides, pour un code exécutable optimisé. En effet, sur un Macintosh de série II, ils traitent 48 000 lignes par minute!

L'éditeur, le débogueur, le gestionnaire de projet et les supports ont également les mêmes caractéristiques qu'en pascal, mais avec de nombreuses améliorations. Le Think C se distingue par la qualité des bibliothèques de classes, Mac, Ansi et Unix, livrées avec les sources, et par le générateur en ligne de code assembleur.

Les blocs de construction fournis contiennent tous les éléments de l'environnement Macintosh, menus, fenêtres et contrôles, et facilitent, par l'ajout de sous-classes, la personnalisation des applications.

La version 5.0 vient d'être commercialisée aux Etats-Unis, et, spécialement conçue pour le Système 7, devrait bientôt être disponible sur le marché européen.

Think C

• Plate-forme de développement orientée objet • Système 6 recommandé, sinon version 5 • 1 Mo RAM au minimum • Compilation, épuration et liaison rapide • Editeur multifenêtre et de fichiers • Débogueur et générateur assembleur • Application, accessoires de bureau, pilote de périphérique, ressources • Support Toolbox, segmentation mémoire, MPW Projector, MacApp et routines OS • Bibliothèques Mac, Ansi, Unix et objets •

Le retour du basic

Le Realizer de Within Technology, dangereux concurrent du fameux Visual Basic, fait partie de cette nouvelle génération d'outils de développement professionnels sous Windows, qui suppriment les aspects ingrats du SDK, pour une puissance, une rapidité et une convivialité supérieures.

Le kit se constitue tout d'abord d'une série d'outils programmables, d'un module de développement en basic donnant accès aux ressources objets de Windows mais aussi d'un éditeur de formes graphiques.

Dans un environnement interactif, d'architecture ouverte, l'utilisateur trouvera aussi un vrai débogueur, pourra créer des menus, employer ou définir une aide en ligne et bénéficier d'un système de rapports et d'un contrôle total de fontes et des couleurs. Un gestionnaire de tâches «scheduler» et un générateur d'animations graphiques agrandiront également le champ d'action des professionnels.

Pour le développement en basic, un large éventail de fonctions standards, mathématiques, statistiques, de formats de fichiers et de fonctions d'accès aux applications Windows et DOS sont disponibles.

Le module «run time» autorise la conception des applications autonomes. Les liens et les déclenchements entre les différents outils développés possèdent également leur propre gestion.

Grâce aux échanges dynamiques de données et de bibliothèques, DDE et DLL, ainsi qu'avec l'accès aux fonctions API, Realizer peut rentrer en interaction avec Paradox Engine, SQL Windows, les langages C et pascal.

Le «Formdev» génère des interfaces complètes, d'après les dessins intuitifs de l'utilisateur muni de sa souris, et cela donc sans une ligne de programmation.

Realizer

• Développement complet sous Windows 3 • Equivalent professionnel du Visual Basic • Rapidité, puissance, convivialité • Compilateur basic, environnement interactif, outils programmables, graphes, dessins, tableurs, objets Windows, scheduler, éditeur et générateur d'interfaces • Déclenchements et liens entre applications • Import-export Lotus, Excel, Paradox Engine, SQL Windows, C, pascal •

Re-sources

Graftool 3.3 de 3D Corp.

Ce produit est distribué par Execusoft, c'est un nouveau type de logiciel s'adressant aux professions scientifiques et technologiques.

Contrairement aux tableurs, les relations complexes entre les données sont manipulées directement sur le graphique.

En introduisant de nouveaux paramètres et en modifiant les hypothèses de travail, l'utilisateur peut aisément mettre en œuvre des simulations sans repenser l'ensemble du graphique.

Doté d'une interface de type Windows, Graftools offre une vaste palette de graphiques 2D et 3D, depuis les diagrammes en bâtons jusqu'aux graphes en coordonnées polaires.

Graftools 3.3

• Analyse et manipulation des données à partir d'un graphique • Palette de graphes en 2 ou 3D et de tous les types • Solveur d'équations, fonctions statistiques et mathématiques • Fonctions de lissage, de régression et d'interpolation •

Plusieurs outils permettent de modifier les données de manière interactive. Un curseur intelligent, telle une sonde, donne toutes les informations numériques sur la position active.

Ainsi, ce logiciel est capable d'analyser et de représenter jusqu'à 268 millions de points. Il comprend aussi un solveur d'équations, des fonctions de transformation des données, comme le lissage, la régression, l'interpolation, ainsi que des fonctions statistiques et mathématiques.

Outre ces prestations évoluées, le logiciel gère le paramétrage de la taille, des axes, du texte et des vues. Graftools fonctionne avec au moins le DOS 3, 640 ko de RAM, une carte graphique ainsi qu'un écran couleur. De plus, il est vivement recommandé par le concepteur d'utiliser un coprocesseur arithmétique.

Formation

Avec l'importance croissante des langages de programmation orientée objet, nous souhaitons souligner la commercialisation d'un produit de formation particulièrement convivial.

En effet, Borland, après «Le Monde des objets», propose aux développeurs professionnels un ensemble de cassettes vidéo intitulé «Le Monde du C++». Spécialement conçu pour les habitués du langage C désirant apprendre la programmation orientée objet (POO) à leur propre rythme, ce cours de deux heures se compose de 21 modules.

Chaque leçon traite un aspect différent du C++. Les notions abordées sont toutes illustrées par des présentations graphiques et par des exercices.

Un manuel de 150 pages et une disquette assurent également le support de formation. «Le Monde du C++» est indépendant de la configuration matérielle et logicielle. Il nécessite toutefois un compilateur ou un traducteur compatibles AT&T 2.0.

Multimedia Development kit

C'est un produit édité et distribué par Microsoft. Dans le cadre de la norme MPC, régissant les débuts du multimédia sur PC, une ressource spéciale s'imposait, vu le large éventail d'applications pédagogiques, médicales, scientifiques...

Microsoft met à disposition une série d'extensions, de bibliothèque, de pilotes de périphérique et de formats de fichier, pour Windows, qui est l'environnement choisi pour le MPC.

Précisons d'abord qu'un minimum d'équipements sera de toute façon nécessaire. Le développeur devra ajouter à son ordinateur un kit d'extension matériel et logiciel. Il se compose d'une carte d'acquisition et de restitution sonore, comme la Sound Blaster Pro ou le Pro Audio Spectrum, d'un lecteur de CD-ROM et, bien évidemment, de Windows multimedia.

Par ailleurs, certaines machines , comme le Victor 386 MM, intègrent déjà tous ces équipements (pour plus de détails, consulter notre numéro 78). Le support du son et d'une vidéo étendue à l'environnement Windows implique la présence d'outils de création capables de faire face à la diversité des informations et des standards.

Contrairement à celle du Software Development Kit, l'interface du MDK, nommée «MC I» (interface de contrôle des médias), est plus simple d'usage.

Outre sa convivialité, elle se distingue par la centralisation de toutes les fonctionnalités. En plus des outils logiciels, cela se traduit par de nouveaux formats de fichier. La norme Midi (interface digitale des instruments musicaux) est ici pleinement supportée.

Petite parenthèse, le PC peut ainsi piloter de nouveaux périphériques, comme un clavier électronique, et, par la suite, récupérer leurs données grâce au format «.Mid».

Plusieurs autres types de fichiers génériques sont retenus par Microsoft. Les fichiers «Wave» stockent les informations sonores, tandis que les «Dib» (bitmaps indépendants des périphériques) offrent la possibilité de compression RLE et une meilleure précision des images. De plus, il existe deux nouveaux formats graphiques, le «PAL» pour les palettes de couleurs et le multimedia moovie pour les animations vidéo.

Une nouvelle plate-forme de création sous Windows.

NOUVEAU VERSION 3.0 EN FRANCAIS

...Si programmer est votre métier

Nouveau ! BORLAND C++ & Application Frameworks 3.0 est le nec plus ultra pour les programmeurs professionnels C et C++ et pour leurs applications Windows et DOS. Grâce à des optimisations inégalées, à des outils puissants, à un environnement Windows sans pareil et à des architectures d'application orientées objet, BORLAND C++ & Application Frameworks 3.0 n'a pas d'égal. Si programmer est votre métier, c'est exactement ce dont vous avez besoin.

La POO vous simplifie la vie

BORLAND C++ & Application Frameworks 3.0 simplifie la programmation en vous offrant des objets d'interface qui s'inscrivent directement dans votre application. Héritez automatiquement des fenêtres, des menus, des barres de défilement, du support souris et plus encore. En outre, il suffit d'une seule ligne pour ajouter un Editeur. Avec la programmation orientée objet (POO), vous bénéficiez de code réutilisable, d'une extensibilité et d'une maintenance simplifiée car les applications sont construites sur un code testé et fiable.

De nouvelles fonctionnalités vous offrent d'incroyables options de programmation !

Regardons quelques-unes des fonctions étendues de BORLAND C++ 3.0 :

- ANSI C et C++ 2.1 avec templates
- L'optimiseur global comprend :
  - Allocation globale de registre
  - Sous-expressions communes locales et globales
  - Variables d'induction
  - Optimisation de boucle et de branchement
  - Passage de paramètre par registre
  - Et dix autres optimisations de pointe
- Augmentation de la vitesse de compilation C++
- Environnements de Développement Intégrés Windows et DOS
- ObjectBrowser™ visuel pour voir d'un seul coup d'oeil les relations entre classes
- Le support «DPMI» pour le compilateur et les environnements «EDI» accroissent considérablement la capacité de traitement
- La bibliothèque EasyWin™ vous permet de convertir facilement vos programmes DOS en programmes Windows
- Resource Workshop pour créer visuellement vos interfaces utilisateur Windows
- Compatibilité étendue Microsoft® C
- Utilitaire de recherche de message WinSight™
- Turbo Debugger™ pour DOS et Windows
- Turbo Profiler™ pour DOS et Windows
- Turbo Assembler™ orienté objet

Avec le nouveau BORLAND C++ & Application Frameworks 3.0 vous avez tout cela, avec en plus :

- ObjectWindows™ - l'architecture d'applications Windows
- Turbo Vision™ - l'architecture d'applications DOS
- Le code source de la bibliothèque d'exécution et des architectures d'applications

Optimisé pour les professionnels

BORLAND C++ 3.0 (3 995 FHT)* et BORLAND C++ & Application Frameworks 3.0 (5 995 FHT)* sont adaptés à vos besoins. N'attendez pas. Car lorsqu'il s'agit de programmation professionnelle, il n'y a pas mieux pour gagner sa vie que Borland C++.

Contactez votre distributeur dès aujourd'hui ou appelez le 34.65.60.60

* Prix public conseillé

BORLAND

Leader en Programmation Orientée Objets

43, avenue de l'Europe - BP 6 - 78143 Vélizy-Villacoublay - France
Tél. (33) (1) 34.65.60.00 Télécopie (33) (1) 34.65.38.77 - **Minitel 3616 Borland**

BEAC

Service lecteurs n° 6 (dernière page)

Enfin, le format de fichiers «RIFF» (fichiers de ressources au format interchangeable) regroupe ces diverses données en une même structure de sous-modules indépendants nommés «chunks». Ce format est également supporté par Presentation Manager sous OS/2. En effet, si différents éléments composent une application multimédia, images fixes, animées, son, texte, etc., il faut les gérer dans un format de fichiers composite.

MPC

• Développement sous Windows multimedia • Nécessite un kit d'extension ou une machine à la norme MPC • Interface intégrée, plus conviviale, gérant le son, l'image fixe et animée • De nombreux pilotes de périphériques, CD-ROM, joystick, standard Midi... • Nouveaux formats de fichiers évolués et composites • Outils de préparation des données : son, image, palette de couleurs, conversion et édition multiformat • Bibliothèques et fonctions en C • Aide hypertexte, documentation complète, exemples •

Mis à part un puissant utilitaire de conversion, le programmeur dispose de quatre principaux outils logiciels, destinés à la préparation des données. «BIT EDIT» est un éditeur de fichiers graphiques assurant les bases du traitement de l'image, de ses caractéristiques et des couleurs. Une douzaine de formats est supportée, dont le BMP, le PCX, le TIFF, le PICT et le TGA. «PAL EDIT» assurera la même fonction pour les palettes de couleurs. L'utilisateur peut choisir : ajouter des couleurs, les trier ou en modifier les paramètres de saturation, de luminance, ainsi que les niveaux de RVB (rouge, vert, bleu).

La sauvegarde s'effectue en format PAL. «Wave edit» est l'un des outils-clés du Multimedia Development Kit. Ce logiciel permet de créer et de modifier des fichiers sonores «.wav». Concrètement, l'utilisateur peut enregistrer sur disque une source sonore analogique, connectée à la carte d'acquisition, l'écouter et l'éditer, tout en visualisant la fréquence sonore sous forme d'une représentation graphique dans le temps.

Cela consiste à effectuer des copier, couper ou coller, à changer les paramètres acoustiques, tels le volume ou les caractéristiques d'échantillonnage.

Ces dernières varient de la mono à la stéréophonie, avec un codage sur 8 ou 16 bits. Trois fréquences d'échantillonnage sont possibles : 44,1, 22,5 ou 11, 025 kHz. Un dernier outil, «File Walk», sera particulièrement apprécié des programmeurs. C'est là que toute l'application multimédia sera développée. Ce logiciel est un éditeur supportant l'ensemble des fichiers générés par le multimédia, c'est-à-dire les huit formats précités.

De plus, il permettra de visualiser en hexadécimal tout autre fichier. En ouvrant un fichier générique, RIFF par exemple, l'utilisateur visualise les principales procédures du code source et les offsets.

Pour en connaître le détail, y insérer un module ou un commentaire, il suffit d'un double-clic. La même opération est valable pour descendre l'arborescence.

Une boîte de dialogue apparaît et affiche la bibliothèque des fonctions disponibles avec leur traduction en hexadécimal. A tous ces outils, s'ajoutent ceux propres à Windows multimedia, c'est-à-dire un panel de contrôle extensible, offrant un large éventail de pilotes et de fonctionnalités : mode d'affichage, joystick, Midi, son, économiseur d'écran...

Hormis l'aide logicielle, l'utilisateur dispose de trois documentations, toutes très complètes. La première offre un tour d'horizon des outils destinés à la préparation des données.

La deuxième, «Work Book» décrit l'architecture de Windows multimedia à travers ses fonctions, standards et leur usage.

Enfin, dans la dernière, le programmeur trouvera tous les détails de ces fonctions en C, tels le type et les arguments, dans le guide de référence, avec de nombreux exemples et illustrations.

HP Visual Engineering Environment

Un générateur de systèmes experts L4G édité et distribué par Hewlett-Packard.

Ce nouvel environnement de visualisation de HP en ingénierie, HP VEE, permet aux ingénieurs et aux chercheurs de résoudre les problèmes applicatifs dans un langage plus intelligible que celui de la machine.

Ce produit permet aux professions technico-scientifiques de créer des modèles de conception et de traitement, selon une méthode très intuitive, pour des applications industrielles ou en temps réel.

HP VEE se compose de deux modules principaux. HP VEE - Engine autorise la création des solutions scientifiques ou techniques d'usage courant, tandis que HP VEE - Test assure la gestion d'objets et diverses fonctionnalités destinées au test et à la mesure. Supporté sur les plates-formes HP-UX utilisant X-Window, ce logiciel est employable sur les stations de travail HP 9000 série 300 ou Apollo de série 400.

HP VEE

• Générateur L4G de systèmes experts en ingénierie • Environnement HP - UX • Utilisable sur les stations de travail 9000 / 300 ou Apollo 400 • Organisation d'objets visuels, liaisons • Saisie, analyse et présentation personnalisée des données •

L'utilisateur n'aura plus besoin de connaître les règles de syntaxe, les structures de programme, les types de données ni les autres subtilités techniques. Les objets visuels, telles les icônes, se trouvent reliés dans un organigramme pour représenter le traitement. Ainsi, les fonctions de saisie, d'analyse et de présentation se révèlent plus simples à organiser.

Cela inclut le stockage des données, leur circulation, la modularité des opérations, la mise au point et la génération de documentation. D'autres objets représentent les fonctions de conception les plus fréquentes, comme le

traitement numérique du signal ou les transformées de Fourier, et assurent la mise en forme graphique et textuelle.

High Screen 5.5

La nouvelle version 5.5 du célèbre High Screen est un puissant générateur d'écran, intégrant le support du clavier et de la souris.

High Screen 5.5

• Edité et distribué par PC Soft• PC-Soft • Générateur d'interface • Mode texte et graphiques • Environnements DOS, OS/2, Windows et Unix • Support du clavier et de la souris • Gestion d'événements • Création d'aide • Récupération des sources en basic, cobol, pascal, fortran, C et dBase • Outils de présentation et d'impression • Bibliothèque d'icônes • Plusieurs modes de compilation • Pas de redevance à la diffusion •

Disponible pour les environnements DOS, OS/2 et Windows, il comprend même certaines extensions pour Unix. Les sources sont sauvegardées dans les langages de programmation les plus courants, basic, cobol, fortran, pascal, C, dBase..., avec un début de programmation orientée objet.

Outil de productivité, le logiciel gère les écrans comme de simples fichiers.

Ainsi, l'utilisateur peut sans aucun problème les visualiser ou les fusionner avec d'autres éléments.

Chaque élément de l'interface, zone de saisie, ascenseur, menu, cadre, icône, se traite comme un objet indépendant.

La gestion de ses attributs, couleur, ombrage, taille des caractères, se trouve simplifiée par une modification interactive en temps réel. En ce qui concerne la souris, l'utilisateur peut définir des zones actives, que ce soit sur une image ou du dessin. L'attribution de raccourcis au clavier s'effectuera avec autant de facilités.

Côté graphique, High Screen gère les modes texte étendus ainsi que les CGA, EGA et VGA.

L'utilisateur détermine comment l'interface doit évoluer, au fur et à mesure du traitement, et cela grâce à la prise en considération des divers événements. Cela autorise donc l'annulation d'une commande antérieure.

Outre les possibilités de dessin et de programmation, High Screen offre bien d'autres services, comme la création d'aide en ligne ou magistrale, la visualisation des tables de caractères ou le support d'un deuxième moniteur pour tester l'interface.

Lorsque tout est au point, un dernier outil permet d'imprimer les dessins d'écran, en indiquant les coordonnées propres à chaque objet. Si High Screen ne requiert que 256 ko, la mémoire étendue est supportée.

Plusieurs modes de compilation s'avèrent possibles : soit il y a séparation entre le programme principal et un module résident et d'autres fichiers, soit tous les éléments sont regroupés, dans un seul exécutable par un système de liens.

L'utilisation et la commercialisation du logiciel ne réclament pas de redevance.

SQL Windows 3.0

SQL pour Windows permet une utilisation conviviale des bases SQL

SQL Windows est un outil destiné aux développeurs d'applications graphiques pour les bases de données SQL. Cette version présente deux nouveautés :
- une orientation objet,
- une meilleure gestion du travail d'équipe.

Dans un premier temps, l'utilisateur a la possibilité de créer l'interface de l'application, dans un module nommé «Design Windows» qui offre une palette d'objets personnalisables, dont les images, et cela, sans les contraintes de SQL Form. En effet, la souris représente ici le périphérique de base.

L'utilisateur choisit les attributs de chaque entité dans le «Customizer», boîte de dialogue gérant le type de données, leur couleur et leur fonte de caractères. Parallèlement, l'«outliner» donne dans une deuxième fenêtre le code formaté correspondant.

Les bibliothèques de type «include» faciliteront les développements de longue haleine, grâce à une meilleure gestion de projet. Cela se traduit par une modularisation de l'application, avec des ressources partageables selon plusieurs niveaux de sécurité.

On peut observer, par exemple, que la modification d'une variable dans un «include» entraîne une mise à jour automatique de l'application.

Quant au langage de requête, plus de 100 fonctions SAL (SQL Windows Application Language) sont disponibles pour la gestion de tables, de listes, de fichiers et de répertoires.

De plus, Report et Express Windows assurent une meilleure capacité d'édition, que ce soit des rapports ou des formulaires, et permettent la création de prototypes logiciels.

Le premier intègre une palette d'outils, le support d'états multicolonnes, ainsi que de nombreuses options de mise en pages et de présentation. Le second génère des applications totalement compatibles avec SQL Windows.

Bientôt disponible en français, le produit existe en plusieurs versions, dont une mise à jour du modèle 2.0 ainsi que le Starter Kit client-serveur incluant Quest et SQL base.

SQL Windows 3.0

• Développement d'applications graphiques pour les bases SQL • Approche orientée objet • Création d'interface, palette d'objets • Bibliothèques modulables et partageables • 150 nouvelles fonctions au service du langage de requête • Modules de reporting évolué • Générateur de prototypes • Edité et distribué par Gupta Technologies •

Mac Irma Api

Il s'agit ici d'un produit permettant le développement de passerelles entre les Macintosh et les sites centraux IBM

Pour la communication entre les Macintosh de séries SE et II et les sites centraux IBM, DCA commercialise depuis peu un nouvel outil de programmation.

Compatible avec les

réseaux locaux CUT et DFT, Mac Irma permet de développer des applications personnalisées au moyen de langages évolués tels le C et HyperTalk.

Mac Irma Api

• Edité et distribué par DCA • Kit de développement réseau • Passerelle Macintosh - site central IBM • Grande compatibilité matérielle et logicielle • Module d'interface de langage • Support des langages structurés courants • Nombreux exemples fournis •

Cette plate-forme de développement fonctionne avec une carte de communication du même fabricant et peut être utilisée avec tous les logiciels de la gamme Mac Irma et Mac Irma LAN. Ces derniers gèrent les configurations monopostes ou en réseau local.

Il faut, par ailleurs, noter que Mac Irma est compatible avec un sous-ensemble d'IBM, EEHLLAPI version 1.2.0 pour les environnements CUT en monosession, DFT en multisession, les émulateurs en réseau local et les fonctions 3270 d'EEHLLAPI.

En conséquence, les anciennes applications, écrites sur PC, sont aisément portables sur Macintosh, et les nouvelles peuvent être développées selon un code de communication identique dans ces deux univers.

Les fonctions d'envoi et de réception de fichiers de Mac Irma Api sont compatibles avec IBM Ind$file dans les environnements hôtes VM/CMS, MVS/TSO et CICS. Le logiciel possède un module d'interface de langage (LIM) pour les principaux langages de programmation, dont HyperCard, C, MPW, pascal, Think Pascal. Cette caractéristique renforce encore la portabilité des sources.

Ces modules servent d'interprètes entre les langages et les logiciels d'émulation.

Ce kit de développement est livré avec de nombreux exemples de codes, de scripts et de stacks.

D'autres versions existent pour des passerelles différentes.

Microsoft franchit un pas de géant

Avec l'importance croissante du langage SQL, Microsoft propose une nouvelle version de ce moteur de requête. Il est disponible aussi bien sous DOS, Windows et OS/2.

Par l'intermédiaire du nouveau MS SQL Bridge, les utilisateurs d'Unix, de Macintosh et de VMS peuvent accéder au serveur SQL de Microsoft. La version 4.2 s'exécute sous MS LAN Manager, Novell Netware, IBM LAN Server et Banyan Vines.

Plusieurs améliorations ont été apportées à ce produit, notamment une meilleure gestion des données, compatible avec les cartouches de sauvegarde en ligne.

Microsoft SQL

• SQL version 4.2 • Edité et distribué par Microsoft • Environnement DOS, OS/2, Windows • Extensions pour Unix, Macintosh, VMS • Supporte LAN Manager, Novell Netware, IBM LAN Server, Banyan Vines • Outils d'administration et d'analyse • Partage des données entre serveurs •

De même, le produit supporte les dispositifs de «mirroring», qui sont utilisés pour les tolérances de pannes, ainsi que les déclencheurs en série.

Un utilitaire livré en standard avec SQL Server améliore la configuration du système et assure une exploitation plus confortable pour l'administrateur du réseau. S'ajoutent à cela les «distributed data management facilities». Ces outils permettent la gestion distribuée des données, par l'appel de procédures reliant un serveur à un autre.

Quant aux «open data services», autre nouveauté, ils associent les sources de données externes et les systèmes d'information pour une meilleure analyse.

Microsoft introduit également un nouveau concept : les «scrollable database cursors» qui permettent de se déplacer plus aisément à travers les données et facilitent le développement graphique sous Windows.

Du nouveau pour SAA

La nouvelle version du langage de commande de SAA, Personal REXX 3.0, est enfin disponible sur PC. Cet interpréteur permet de développer des applications structurées, portables sur les systèmes VM/CMS et MVS/ESA.

Il dispose a cet effet d'un gestionnaire d'écran, d'un module résident qui exécute des procédures REXX à l'image des fichiers de traitement par lots du DOS (les fameux «Batch»).

Il comporte également un device driver destiné aux simulations de périphériques de sortie dans la pile.

Personal REXX peut être interfacé avec d'autres langages (langage d'assemblage, pascal et C), avec partage des variables.

Personal REXX

• Version 3.0 • Edité par Quercus Systems • Distribué par Danish International • Interpréteur de langage REXX • Environnement PC • Compatibilité VM/CMS et MVS/ESA • Générateur d'écran, pilotage de fichiers de type Batch, interface assembleur, pascal et C • Prise en compte des ressources matérielles (reconnaissance de la mémoire paginée EMS) et logicielles (accès aux répertoires et fichiers du DOS) •

Il est reconnu par KEDIT comme un langage de macro commandes, ce qui donne la possibilité de construire des «applications» en testant les codes de retour.

Il dispose également des fonctions de trace classiques, prend en compte toute la mémoire étendue de type LIM EMS 4.0 et intègre aussi des fonctions «PC» d'accès au matériel et au DOS.

Enfin, il effectue des accès à la pile à l'aide des commandes «Push», «Queue» et «Pull».

La compatibilité CMS/VMS est certainement l'un des aspects les plus intéressants de REXX, autour de laquelle s'articule des commandes telles que «Makebuf», «Dropbuf», «Desbuf», et «Sentries».

Re-sources

L'éditeur professionnel

La version 4.0 de Kedit intègre un éditeur de texte plein écran qui reprend les fonctions de Xedit, l'éditeur standard du système IBM VM/CMS pour mainframes, S370 et S390. L'affichage en mode VGA est désormais supporté.

A noter qu'il est possible de générer des fichiers ASCII purs, c'est-à-dire sans les habituels codes de contrôle que l'on retrouve généralement dans les éditeurs de textes évolués. Kedit est capable d'ouvrir un fichier de n'importe quelle taille : la seule limite provient de la place disponible en mémoire conventionnelle ou bien en mémoire paginée LIM EMS 4.0.

L'éditeur donne accès à des commandes étendues telles que la recherche et le remplacement selon des critères de lignes ou de colonnes. Les manipulations de blocs font appel à la même logique.

Des commandes génériques permettent pour leur part de réaliser des copies, déplacements ou effacements d'une ligne ou d'un groupe de lignes. A l'image de Word, l'affichage peut être sélectif : l'utilisateur peut visualiser tous les codes, certains d'entre eux ou aucun.

Des macro commandes font, bien entendu, partie intégrante de Kedit et sont compatibles avec KEXX ou Personal REXX.

L'utilisateur peut personnaliser le chargement du Kedit à l'aide du fichier de ressources

Profile.ked qui joue le rôle d'un Autoexec.bat détermine la couleur des différentes zones d'affichage, les préfixes associés aux fichiers, etc. L'utilisateur a également la possibilité de programmer les douze touches de fonction du clavier étendu ainsi que les combinaisons obtenues avec les touches Alt Ctrl et Shift.

Dernière caractéristique importante de Kedit : il peut éditer simultanément plusieurs fichiers par le biais de fenêtres multiples.

KEDIT

• Version 4.0 • Edité par Mansfield Software Group • Distribué par Danish International • Editeur pleine page pour MS-DOS • Reprend les fonctionnalités de Xedit, supporte l'écran VGA • Edition des textes purs, taille des fichiers illimitée, reprogrammation des touches du clavier, ouverture simultanée de plusieurs fichiers, affichage multi fenêtré, etc. •

SÉLECTIONNÉE PAR **MICRO SIGMA**

LA LIBRAIRIE GRAPHIQUE **HALO Professional**

RÉPOND AUX EXIGENCES DE VOS APPLICATIONS GRAPHIQUES

- Support de nombreux **périphériques** (écran, imprimante, table traçante, souris...)
  Intégration de leurs évolutions successives
  Utilisation optimale de leurs fonctionnalités
- Disponibilité pour les **langages** C, Pascal, Fortran, Basic (mode réel), Assembleur, Ada.
- Utilisation en DOS mode réel et **mode protégé** grâce à la compatibilité avec les DOS Extenders (Rational System, Phar Lap, Ergo).
  Allocation dynamique de la mémoire : pas de module résidant, chargement du driver nécessaire lors du run-time.
  Rapidité d'exécution.
- Nombreuses fonctionnalités graphiques :
  - Diversité des primitives graphiques
  - Fenêtrage avec clipping
  - Polices de caractères vectorielles et bit map avec éditeurs
- Support technique en France assuré par **MICRO SIGMA** , distributeur agréé Media Cybernetics
- A partir de 3 950 Frs H.T.

Consultez-nous au 46 22 99 88 Fax : 47 63 84 42
Recevez une **documentation** par **FAXFACTS** au 47 63 59 93 - HALO = N° 317

MICRO SIGMA Outils de Développement
42, rue Legendre - 75017 PARIS

HALO Professional est une marque déposée de MEDIA CYBERNETICS

Service lecteurs n° 7 (dernière page)

DEVELOP

Version 100% linkable!

HIGH SCREEN 5.5

Générateur d'écrans
- DOS, WINDOWS, OS/2, UNIX.
- Tous langages (oui, tous!)

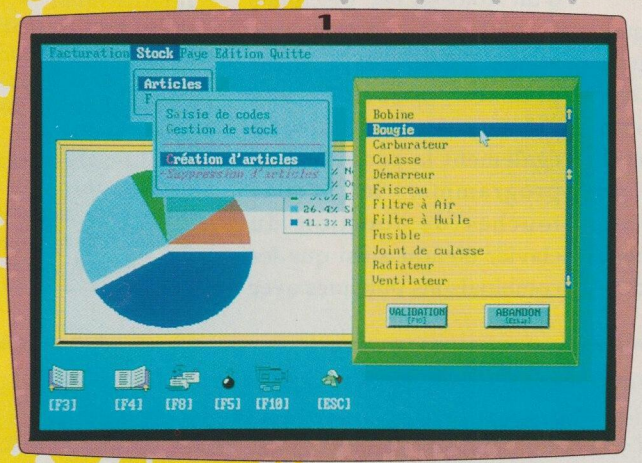
Nouvelles versions:
- **UNIX**
- **Windows**

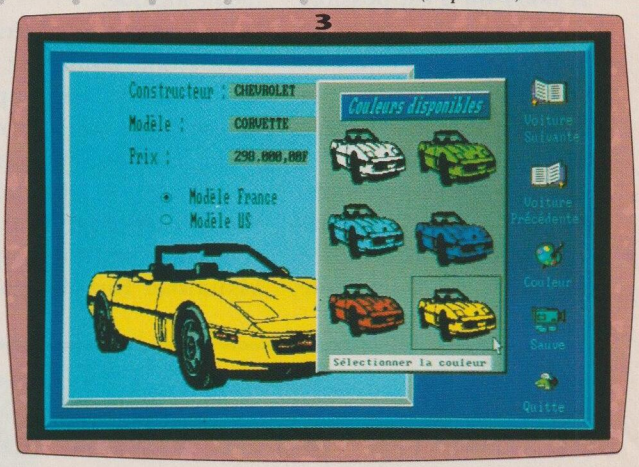
1- Menus déroulants, fenêtres, saisies, icônes, graphiques,... : tout High Screen 5 en un écran !

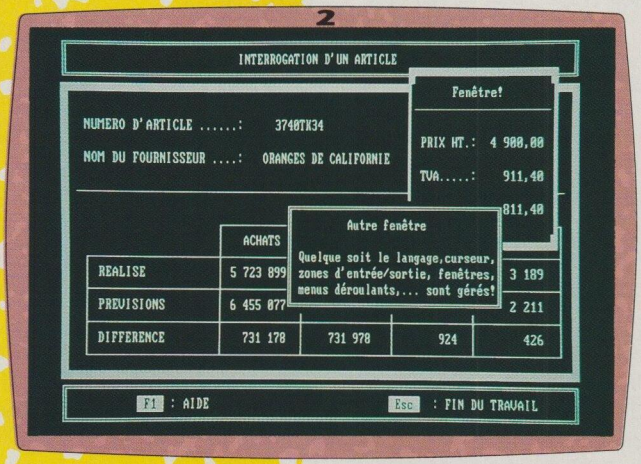
2- High Screen fonctionne sur tous les écrans...

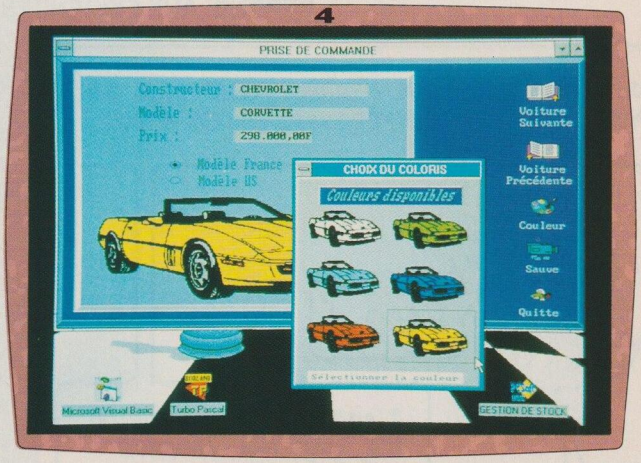
3- Une application graphique sous DOS...

4-... la même sous Windows (compatibilité !)









Que vos applications soient en mode texte ou graphique, au clavier ou à la souris, avec menus déroulants, avec des icones, avec ou sans fenêtres,
High Screen 5 vous permettra de développer 3 à 10 fois plus vite.

La programmation est très simple, avec des ordres en français.
De nombreux utilitaires sont livrés : éditeur d'icones, de dossiers, maquettage,... Tout devient possible, dans tous les langages!
(C, Basic, Pascal, dBase, Nantucket, Cobol, Fortran,...)

High Screen 5 a déjà changé la vie de milliers de développeurs : essayez-le !

A partir de
4900 F HT
5 811,40 F TTC

Disquette de démonstration disponible : appelez !

Nos produits ont reçu de nombreuses récompenses :

HYPER PACK — OUTILS DE DEVELOPPEMENT — PRIX D'EXCELLENCE — MICRO SYSTEMES

Meilleur outil de développement — SOFT & MICRO

LABO QUALIFIE — INFO PC

LE SUPPORT TECHNIQUE
Un des points forts de PC SOFT est le support technique, efficace, disponible et... sympathique.
Notre équipe de techniciens chevronnés est à votre disposition. Gratuitement.

GARANTIE "ESSAI SANS RISQUE"
Si le produit ne vous convenait pas pour une raison quelconque, vous pourriez nous le retourner (en état d'origine bien sûr) dans les 10 jours suivant sa réception pour un remboursement intégral. Conditions au dos du tarif.

Malgré le soin apporté à la rédaction de ce document, les informations ne sont pas contractuelles. Toutes les marques sont déposées par leur propriétaire respectif.

PEZ VITE

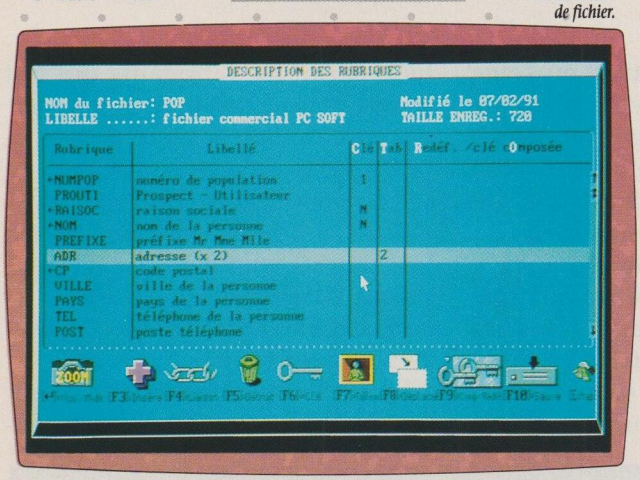
Version 100% linkable!

HYPER FILE 3

SGBD Gestionnaire de fichiers

- DOS - WINDOWS - OS/2
- C - Basic - Pascal

Création de structure de fichier.



Hyper File 3 est le plus rapide des SGBD. Après avoir défini facilement la structure des fichiers, leur utilisation dans les programmes est très simple.
La programmation (en C, Basic, Pascal) s'effectue en français.
De plus Hyper File 3 édite automatiquement le dossier d'analyse.

De nombreux utilitaires sont livrés.

Si vous manipulez des fichiers, testez Hyper File !

A partir de
4900 F HT
5 811,40 F TTC

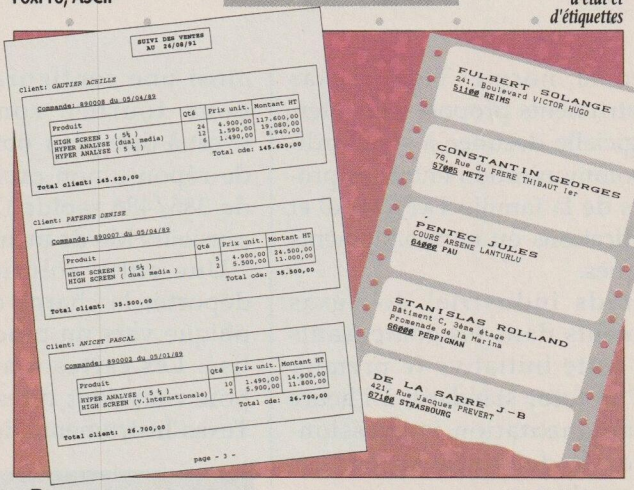
Disquette de démonstration disponible : appelez !

HYPER PRINT 3

Générateurs d'états et d'interrogations

- C, Basic, Pascal, dBase, Nantucket, FoxPro, ASCII

Exemple d'état et d'étiquettes



Réaliser des états imprimés de gestion en quelques minutes seulement, c'est possible avec Hyper Print 3.
Hyper Print 3 rapproche jusqu'à 9 fichiers par état et sait calculer.
Le résultat s'édite sur imprimante (matricielle ou laser) ou à l'écran, ou même dans une feuille de tableur.

Un produit à évaluer d'urgence !

A partir de
4900 F HT
5 811,40 F TTC

Disquette de démonstration disponible : appelez !

HYPER PACK DEVELOPPEUR

Atelier de Génie Logiciel

C - Basic - Pascal

Ce "pack" regroupe High Screen, Hyper File et Hyper Print qui utilisent le même dictionnaire de données.
Le "PACK" permet de développer de superbes applications rapides, très très vite !

A partir de
9900 F HT
11741,40 F TTC

PC SOFT est Fournisseur Officiel de la Préparation Olympique

PC SOFT

LIVRAISON RAPIDE
En France métropolitaine, livraison en 24 heures par transporteur.

~~DISQUETTE~~ DE DEMONSTRATION DISPONIBLE

DOCUMENTATIONS CLAIRES EN FRANCAIS

PAS DE REDEVANCES

SIEGE MONTPELLIER
Tél. (16) 67.032.032
216, rue des Escarceliers, BP 3019
34034 Montpellier Cedex 01
Fax: (16) 67.03.07.87
Support Technique: (16) 67.03.17.17

PARIS
Tél. (1) 48.01.48.88
34, Boulevard Haussmann
75009 Paris
Télex: 290.266 F (MBI)
Minitel: 3614 code PCSOFT

Service lecteurs n° 20 (dernière page)

SYSTÈMES

TIGA
L'affichage graphique haut de gamme selon Texas

ATTA OLOUMI

Avec la norme Tiga, Texas Instruments propose une interface logicielle destinée à standardiser la communication entre les processeurs de la famille TMS 340x0 et les applications ou environnements graphiques.

Le poids industriel de Texas Instruments dans les composants fait de cette initiative le premier jalon d'un futur standard, à l'heure d'une augmentation impressionnante de la demande en performance graphique.

Le mode graphique «consomme» considérablement plus de temps processeur que le mode texte : l'unité d'information de base étant le pixel, le moindre caractère affiché découle d'un processus qui doit prendre en compte le type et la taille de celui-ci, la nature de l'écran, la définition de l'interface graphique, etc.

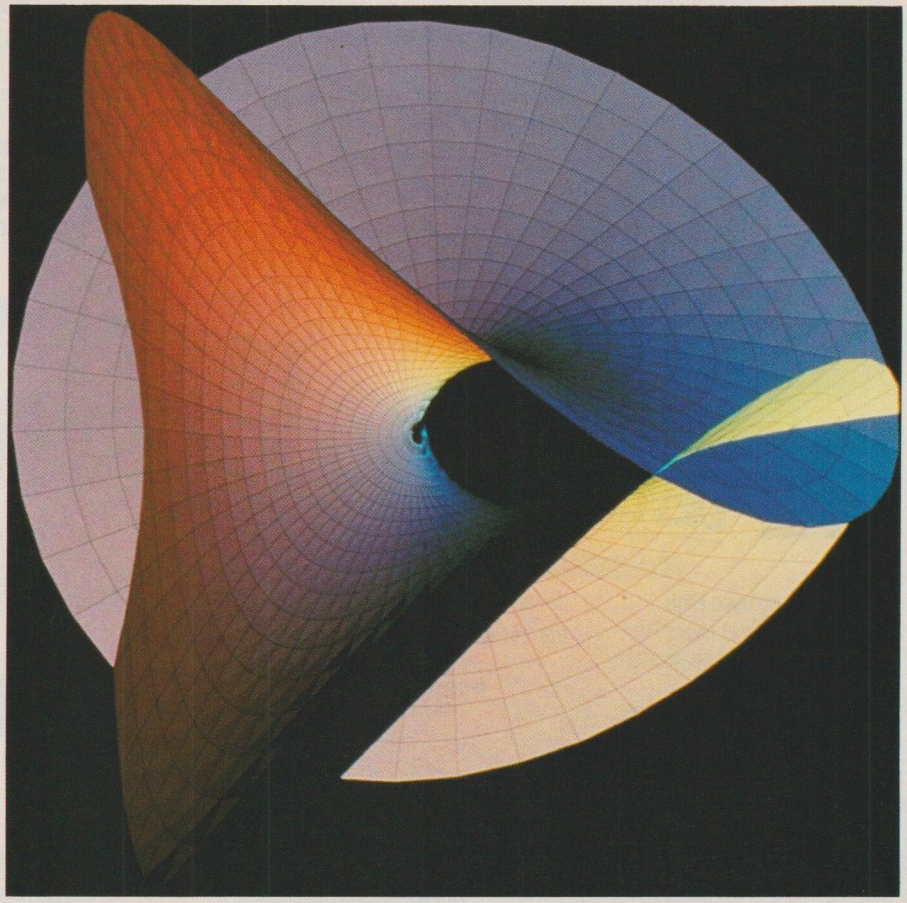
Les séries de points formant les figures géométriques sont individuellement spécifiées à chaque rafraîchissement d'écran.

Contrairement au mode texte, l'affichage ne se résume plus à un transfert de données entre la mémoire centrale et la mémoire vidéo qui contient la liste des pixels. Le mode graphique réclame de longues séquences de calcul car il implique la traduction de formes géométriques traitées comme des équations mathématiques en un ensemble de points.

Les environnements graphiques engendrent une inflation de la puissance requise. Windows 3 demande ainsi une configuration musclée pour tourner convenablement (c'est-à-dire pour obtenir un temps de réponse non prohibitif), à base de 386/486 rapide (25 MHz) et de plusieurs Mo de mémoire vive.

Une autre solution consiste à déporter la charge de travail graphique vers un processeur spécialisé. Les cartes d'affichage ainsi pourvues sont dites «accélérées». Texas Instruments figure en tête de liste parmi les fabricants qui ont commercialisé des processeurs graphiques. Comme à l'accoutumée, la présence d'un environnement de développement apporte une plus-value au composant. Texas Instruments fournit ainsi une bibliothèque de primitives qui s'interfacent entre les applications et le processeur graphique.

La norme Tiga (Texas Instruments Graphics Architecture) est une solu-



Les résolutions autorisées par les processeurs Texas permettent d'obtenir des graphiques de qualité.

L'offre de Texas Instruments recouvre désormais tous les éléments d'une carte graphique avec, en prime, une librairie facilitant le travail du développeur (Tiga) et avec des processeurs haut de gamme délivrant plus de 40 Mflops. Ces derniers, combinés à la bibliothèque de primitives Tiga, comblent les attentes des environnements graphiques tels que Windows et des applications les plus exigeantes dans la CAO 3D, dont la figure de proue demeure AutoCAD.

tion globale pour les cartes graphiques haut de gamme. La mise en œuvre de la carte accélérée passe par l'utilisation de la librairie Tiga, dont les routines sont accessibles à partir de divers compilateurs C. Les aspects matériels demeurent transparents pour le programmeur.

La prise en charge par les primitives Tiga d'une partie de la gestion de l'écran, alors impartie à un environnement graphique donné, augmente très sensiblement la vitesse d'affichage, pour la simple raison que le processeur principal se voit déchargé de ce travail au profit du processeur graphique.

Les primitives Tiga

La librairie Tiga se partage en treize groupes de primitives :

❏ Les fonctions d'initialisation. Avant qu'une fonction Tiga ne soit appelée, il faut initialiser l'environnement graphique et le restaurer à la fin de la session.

❏ Les fonctions d'effacement. Ce sont elles qui assurent la prise en charge du nettoyage de l'écran ou du «frame buffer».

❏ Les fonctions de contrôle des attributs graphiques. Elles permettent de déterminer ou de modifier l'état des attributs graphiques, à savoir la couleur de fond, la position d'un point par rapport au point d'origine d'une fenêtre, la taille d'un outil de dessin, la transparence ou la possibilité de tracer à l'extérieur de la fenêtre de travail graphique. Toutes ces fonctions définissent donc le contexte de la fenêtre de travail.
Exemple : la fonction «cpw(x,y)» (Compare Point to Clipping Windows) retourne une valeur sur 4 bits qui indique la position relative du point de coordonnées x et y, par rapport à la fenêtre de travail (la valeur 0000 indiquant l'appartenance du point à cette dernière).

❏ Les fonctions liées à la palette de couleurs, utilisées pour son initialisation ou son chargement.

❏ Les fonctions de dessins géométriques, tels que les arcs de cercle, d'ellipse, les points individuels, les cercles, les ellipses, les rectangles et polygones. Les contours peuvent être représentés seuls mais les différentes figures sont également disponibles remplies d'une couleur.
Exemple : draw_rect(u,v,x,y) dessine à l'écran un rectangle spécifié par les points x et y pour le coin supérieur gauche et par u et v pour la largeur et la longueur.

❏ Les opérations de déplacement des images rectangulaires «bitmap». Elles consistent essentiellement en des transferts entre une zone source et une zone destination. La fonction «encode_rect» compresse une image bitmap contenue dans un rectangle donné. L'algorithme utilisé est peu élaboré puisqu'il s'agit d'une analyse de chaque ligne horizontale dans laquelle les séquences répétitives de pixels sont remplacées par le nombre de points identiques. Symétriquement, «decode_rect» restaure l'image initiale.

❏ Les fonctions de textes agissant sur le choix des fontes ou retournant le code ASCII des caractères.

❏ Les fonctions liées au curseur, déterminant l'état (par exemple, visible ou invisible) et la position du curseur.

❏ Les fonctions utilitaires offrant la possibilité de connaître la valeur d'un pixel («get_pixel»), d'obtenir des informations sur l'espace travail («get_wksp») ou de convertir une adresse bidimensionnelle x-y en adresse linéaire(«cvxyl»).

❏ Les fonctions de gestion des blocs mémoires permettant, par exemple, d'assigner un bloc mémoire à une fonction.

❏ Les fonctions de transferts de données entre la mémoire centrale et les espaces mémoires du TMS 340x0.

❏ Les fonctions étendues.

❏ Les fonctions d'interruption.

C à l'honneur

Les éléments de la librairie Tiga sont appelables à partir d'un compilateur C ou de l'assembleur. Actuellement, les compilateurs compatibles sont le Microsoft C Compiler 5.0, MetaWare High-C Compiler 1.5 et le Microway NDP C-386 Compiler 2.0. Seul le produit de Microsoft doit se contenter d'un fonctionnement en mode réel DOS. Les deux autres outils de développement acceptent le DOS Extender de PharLap. L'offre assembleur se

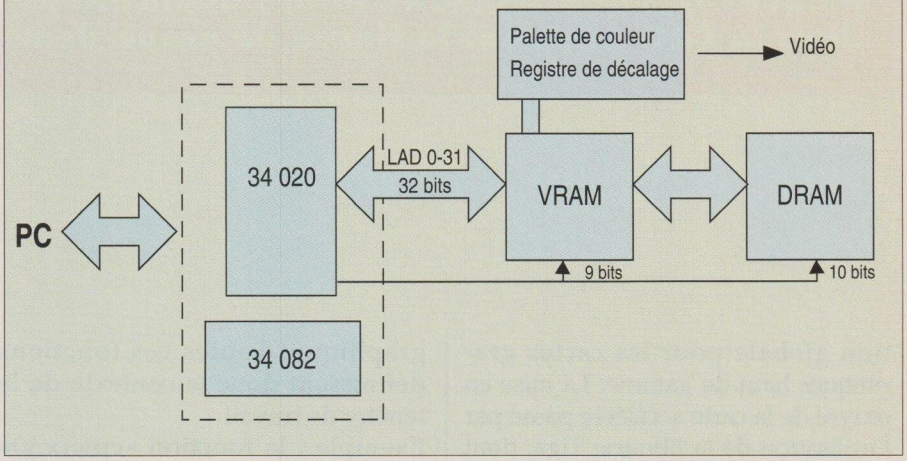
limite, en revanche, au Microsoft Macro Assembleur 5.0 (et au Turbo Assembleur de Borland). Il faut, cependant, veiller à vérifier différentes conventions graphiques. Ainsi, le système de coordonnées de l'écran localise l'origine dans le coin supérieur gauche. Le problème de l'appartenance d'un pixel à une figure géométrique se solutionne par la présence ou pas du centre du point à l'intérieur du périmètre de la figure.

Les coordonnées d'un point ne désignent pas son centre mais la partie supérieure gauche de l'emplacement qu'il occupe. Parmi les constituants de la librairie Tiga, les nombreuses fontes fournies répondent à la quasi-totalité des exigences. Il est possible d'enrichir le choix proposé par ses propres créations. On notera que ces fontes, bien que disponibles en plusieurs tailles, sont de nature «bit map» et non vectoriels.

Leur édition sur support papier, avec une définition supérieure à celle de l'écran fait donc appel à une étape d'interpolation. Cette opération demeure peu coûteuse en termes de calcul quand on la compare à l'interprétation de caractères définis vectoriellement. Sur un écran de 640 par 480 pixels, la taille des fontes peut varier de 5 à 72 pixels. Il ne faut pas perdre de vue la désignation initiale de l'environnement logiciel Tiga qui ne trouve son sens que dans l'interfaçage entre les applications et les processeurs de la famille TMS 340x0.

Chaque élément de la librairie s'exprime comme une suite d'instructions de l'assembleur spécifique TMS 340x0 orienté graphique. La performance se trouve directement en rapport avec la vitesse d'exécution des composants et la qualité de leur implémentation.

Schéma général d'une carte TIGA



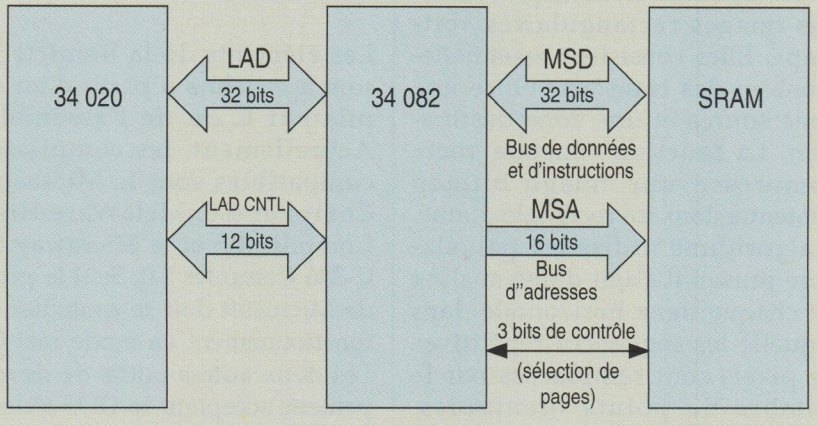
Les processeurs graphiques

Le terme générique de GSP (Graphical Signal Processor) a été choisi par Texas Instruments pour désigner les processeurs graphiques. Ceux-ci se consacrent aux tâches d'affichage grâce à une série d'instructions adaptées aux traitements des pixels et des figures géométriques.

La première génération de GSP, le TMS 34010, a été introduite au début de l'année 1987. Muni d'un bus externe sur 16 bits, ce composant disposait à l'origine d'un cycle de base de 200 ns (soit 5 MHz de fréquence de travail). Les dernières versions atteignent 130 ns. Mais les limitations de ce produit ne lui permettent pas de répondre aux exigences de nombre d'applications. Parmi les défauts rédhibitoires, l'architecture sur 16 bits entiers alourdit fortement la charge de travail lors de la représentation de figures géométriques usuellement codées sur 32 bits flottants dans les logiciels de CAO. De plus, l'absence d'une liaison avec un éventuel coprocesseur arithmétique diminue ses potentiels futurs.

Le TMS 34020 comble les lacunes précédentes et répond aux nouveaux besoins exprimés. Ceux-ci se focalisent sur trois catégories de produits : les logiciels de CAO, les produits de PAO et les environnements graphiques. Chacun d'eux réclame des fonctionnalités propres. Parmi les logiciels de CAO, il faut

La connexion 34 020 / 34 082



distinguer ceux travaillant en 2D et ceux en 3D.

Un plan est constitué d'un ensemble de coordonnées de points codés sur 32 bits flottants et d'une liste de figures reliant les points (polygones, segments, arcs de cercle ou d'ellipse, etc.). La fonction zoom (et plus généralement le redessin d'écran) est le gros consommateur régulier de calculs. Un GSP peut compenser cette carence grâce aux opérations de dessin vectoriel microcodées dans la structure du composant. L'expérience montre que la solution à base GSP convainc car nombre d'utilisateurs d'AutoCAD, référence dans le monde de la CAO en 2D, éliminent en grande partie la latence dans l'interactivité du logiciel par l'adoption d'un contrôleur accéléré.

La configuration 386SX/16 avec coprocesseur, associée à une carte graphique munie d'un GSP haut de gamme, l'emporte en termes de performances sur les 486 à 50 MHz avec une carte d'écran classique. Les logiciels de CAO 2D se satisfont par conséquent des microprocesseurs actuels, sauf pour les fonctions de redessin d'écran.
Cela n'est plus vrai pour les logiciels 3D qui nécessitent des stations de travail haut de gamme en raison de la complexité des calculs. Le cas des logiciels de PAO est plus complexe.

Les différents caractères en mode Wysiwyg doivent être interprétés vectoriellement, et les fonctionnalités associées aux images imposent du traitement «bitmap». Mais le problème du placement et de la visualisation des fontes est lié à la couche sous-jacente, à savoir l'environnement graphique.

Le mode graphique utilise à la fois les capacités vectorielles et «bitmap» du GSP. Afin de répondre à ces demandes, le TMS 34020 travaille sur 32 bits et voit le nombre de ses instructions et modes d'adressage largement augmenté.

Au chapitre des possibilités d'expansion, la présence d'un coprocesseur arithmétique offre un nouvel ordre de grandeur dans les capacités. Les éléments de la librairie Tiga se traduisent en un nombre limité d'instructions du 34020. Le haut niveau des instructions rapproche cette norme du composant.

Le jeu d'instructions du TMS 34020 supporte huit modes d'adressage : immédiat, absolu, direct, indirect, indirect en X Y, indirect avec déplacement, indirect pré-incrémenté et indirect post-incrémenté. Hormis l'adressage indirect en X Y, tous les autres font figure de grands classiques et dépassent le cadre de cet article.

Une adresse XY est bien plus adaptée à la manipulation d'images que le mode d'adressage linéaire. Lors de cet adressage particulier, un registre interne du TMS 34020 sur 32 bits divisé en deux parties de 16 bits est spécialisé pour l'affichage de blocs de pixels. D'où la prolifération des instructions de type MOVE. Les opérations classiques, arithmétiques, logiques, de comparaison, de rotation et de décalage se

LA GUERRE DES COPROCESSEURS ARITHMÉTIQUES

Les interfaces graphiques et les outils de bureautique se contentent d'un processeur de type TMS 34020. La CAO ou le traitement d'image récusent toute solution qui ne prend pas en compte le calcul flottant, aussi bien sur PC que sur une station de travail. Dans la catégorie de l'offre «virgule flottante», le TMS 34082 est en concurrence avec l'Intel i860 (50 MHz), mais également avec les DSP 32 bits flottants, que ce soit le Motorola 96002 (40 MHz) ou même le Texas Instruments TMS 320C30 (33 MHz). Tous ont un multiplieur et un ALU câblés, mais ceux des DSP ne sont que sur 32 bits flottants tandis que le TMS 34082 et le i860 travaillent aussi en double précision sur 64 bits flottants. Cependant, peu d'applications réclament réellement la double précision. Les logiciels de CAO ou de traitement d'image n'ont pas encore atteint les limites du 32 bits flottants. L'autre élément de différence entre ces composants et les coprocesseurs classiques réside dans l'architecture pipe-line qui permet de débiter un résultat par temps de cycle instruction. Les coprocesseurs des RISC de deuxième génération (PA-RISC de HP équipant la série 700, POWER d'IBM au cœur des stations RS/6000 et 29050 d'AMD, ainsi que les futurs RISC de Mips, Motorola ou Sun) reprennent également cette structure. Les instructions MULAC à trois opérandes, qui enchaînent une multiplication et une addition en un temps de cycle, caractérisent les DSP et leur permettent d'effectuer deux opérations à virgules flottantes par cycle d'instruction. Le i860 reprend cet atout contrairement au TMS 34082. La constitution de chaque composant révèle également des différences dans leur mise en œuvre. Le i860XP (à 50 MHz) comprend une unité à virgule flottante (FPU) dont la particularité est une unité graphique 3D câblée (Z-Buffer), une mémoire cache de 16 ko pour les données, une autre de même capacité pour les instructions et un cœur RISC 32 bits entiers. La mémoire cache ne peut être utilisée comme une RAM sans état d'attente. Cette lacune est d'autant plus flagrante que toutes les instructions s'effectuent en un seul temps de cycle. Toutes les fonctions évoluées doivent être émulées dans le programme (même pour les opérations les plus simples comme la division). Le i860XP est actuellement le plus intégré des composants avec 2,5 millions de transistors. Or les deux mémoires caches et leur gestion en occupent approximativement 2,2 millions, le cœur RISC entier moins de 30 000 et le FPU près de 200 000. Mais les mémoires caches ne recouvrent que 60% de la surface, du fait de l'absence de voies de communication comme les bus internes. Le TMS 34082 ne comprend ni RISC entier (ce rôle est dévolu au TMS 34020) ni mémoire cache. Par contre, un microcode contient un grand nombre de fonctions avancées, qui minimisent d'autant la taille du programme, et un bus externe spécifique qui permet l'accès à une mémoire locale sans état d'attente. C'est la raison pour laquelle, et bien qu'appartenant à la même catégorie que le i860, le TMS 34082 intègre moins de 300 000 transistors.

retrouvent dans l'assembleur du composant.

En revanche, l'existence du mode XY implique des instructions particulières telles que ADDXY qui additionne les contenus de deux registres chargés d'adresses en XY. Les appels de sous-routines et autres sauts conditionnels sont également présents.

Les principales instructions

Un grand nombre d'instructions et d'opérations graphiques facilitent la mise en œuvre des applications, d'autant qu'elles sont microcodées afin de tirer parti de l'architecture interne du GSP. Les instructions DRAV et PIXT se consacrent à l'affichage individuel de pixels. La première dessine un point tandis que la seconde transfère un pixel d'une localisation vers une autre.

Une ligne est tracée par l'instruction LINE (pour les connaisseurs, elle fait appel à l'algorithme de Bresenham). Le point final doit être identifié par une adresse XY lors de l'utilisation de l'instruction LINE. La version rapide, FLINE, se distingue par l'adresse linéaire désignant le point final. Le déplacement de blocs de pixels intervient souvent dans les applications graphiques. Ces transferts de groupes de pixels entre différentes régions de la mémoire s'effectuent par le biais de l'instruction PIXBLT (pour PIXel BLock Transfer).

La fonction FILL est plus simplement chargée du remplissage d'une zone par une valeur donnée des pixels contenue dans l'un des registres (COLOR1). Deux instructions spécifiques au TMS 34020 prennent en charge les opérations de recherche et de comparaison de pixels. FPIXEQ recherche un pixel identique à la valeur du registre COLOR0 tandis que FPIXNE réalise l'opération inverse.

Dans les deux cas, la recherche s'interrompt dès qu'un pixel remplissant les conditions voulues est trouvé. Un effort particulier a été fourni lors de l'implémentation de ces deux instructions. De leur célérité dépend la performance du composant puisque ces fonctions sont appelées lors des opérations de remplissage ou de compression de données.

L'ajustement d'une zone de pixels à la taille d'une fenêtre est pris en compte par la fonction CLIP. Enfin, un effet de perspective peut être généré par TFIll qui remplit à l'écran non plus un rectangle mais un trapèze.

Toutes ces instructions particulières capitalisent l'originalité du composant. A la base, elles reposent sur des opérations simples de comparaison et des boucles. Ces dernières ont été optimisées grâce au traitement de la mémoire sous forme linéaire ou de tableau. L'action de la mémoire cache se trouve valorisée car les données se suivent linéairement.

Le TMS 34020 ressemble à un RISC par sa structure interne. Il intègre un pipe-line à trois étages utilisant simultanément le cache, les registres et la RAM. La question du choix d'un processeur dédié aux tâches graphiques se pose avec d'autant plus d'acuité que les performances théoriques des microprocesseurs RISC généralistes, des microcontrôleurs de la dernière génération, des processeurs de traitement du signal (DSP) et des GSP sont pratiquement identiques. Tous bénéficient des apports du pipe-line dans l'architecture interne. Cette structure permet d'effectuer une instruction par temps de cycle. Par rapport aux autres solutions, le TMS 34020 se distingue sur deux points :

❏ Les opérations de comparaison, de recherche ou d'addition ont été en partie câblées.

❏ Le microcode, en ROM interne, contient les instructions graphiques.

Ainsi, les fonctionnalités spécifiques du GSP telles que l'adressage XY, le déplacement ou le remplissage des ensembles de pixels se déroulent plus rapidement que si elles étaient simplement émulées

STRUCTURE DES CARTES GRAPHIQUES À BASE DE TMS 340x0

Outre le TMS 34020 et son coprocesseur, une carte graphique reçoit une mémoire contenant un programme en local et une mémoire vidéo. Une taille mémoire d'un million de mots de 32 bits répond à la plupart des besoins. Tous les mots de cette mémoire sont individuellement accessibles par une adresse linéaire sur 10 bits.

Le bus d'adresses RCA est alors mis à contribution. En revanche, l'adressage XY convient mieux à la mémoire vidéo. A cette fin, Texas propose le TMS 44C251 qui est une RAM vidéo spécifique de 1 mégabit (4 x 256 ko) utilisant deux adresses successives sur 9 bits.

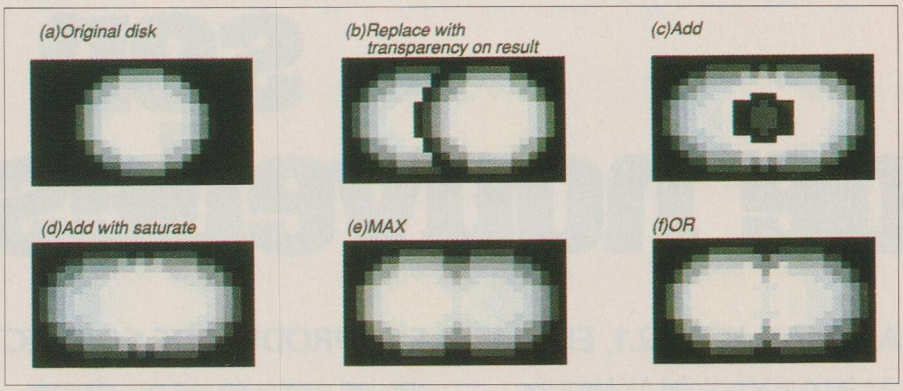
Le TMS 34020 émet ce type d'adresses à travers le bus RCA après multiplexage. Le contenu de la mémoire vidéo (VRAM) génère l'écran. Pour cela, une dernière étape d'interfaçage avec la sortie vidéo est nécessaire.

Le circuit TLC 34075 Video Interface Palette, relié directement à la VRAM, se charge de cette fonction. Ce composant est compatible VGA et reprend la sortie vidéo RGB standard, RS-343A.

Exemples d'opérations arithmétiques et booléennes sur les pixels.

dans des composants généralistes. Il demeure difficile de quantifier le gain par l'emploi du GSP face aux possibilités offertes aux concepteurs de cartes : les contrôleurs d'écran accélérés tirent leur puissance d'un chip sur mesure, d'un processeur RISC généraliste ou d'un GSP.

Le premier cas se retrouve le plus souvent sur des cartes dédiées aux interfaces graphiques. Bien adaptés pour répondre à la demande croissante de puissance de calcul des logiciels de bureautique, voire de PAO, ces circuits particuliers restent insuffisants pour les logiciels de CAO. En effet, ce type de circuits ne contient pas de blocs de fonction à virgules flottantes et ne peut recevoir de coprocesseur arithmétique.



Opérations sur l'intensité.

Une puissance crête de 40 Mflops pour le coprocesseur

Texas Instruments n'a pas choisi la voie du compromis avec son coprocesseur TMS 34082 : la puissance crête monte à 40 Mflops en double précision. Davantage qu'un simple coprocesseur (il peut, en effet, fonctionner de manière indépendante du processeur central), il se distingue par la présence de deux bus de données externes sur 32 bits (LAD et MSD) et d'un bus séparé d'adresse MSA sur 16 bits. Grâce à l'existence des bus LAD et MSD, ce composant accède simultanément au processeur principal et à une mémoire locale ultra-rapide de type SRAM d'une capacité maximale de 64 Kmots, limitée par les 16 bits du bus MSA (il est possible de dépasser cette valeur par l'emploi d'un groupe de trois pattes de broches dédiées à la sélection de pages mémoires).

Cette architecture offre la possibilité, si elle est exploitée, d'éliminer le goulet d'étranglement que constitue le passage par le processeur principal pour l'accès aux données. Le TMS 34082 tient sa puissance d'un multiplieur et d'un ALU (sur 64 bits) qui sont câblés.

L'espace de travail immédiat se réduit à 32 registres de 64 ou 32 bits. En dépit de ces performances qui le placent dans le haut de gamme des coprocesseurs arithmétiques, le TMS 34082 rappelle sa vocation graphique par la richesse de ses instructions.

Celles-ci, microcodées pour la plupart dans une ROM interne, utilisent de manière optimale des ressources internes (travail sur les registres internes uniquement et exploitation des opérations câblées). Le panel des fonctions disponibles est des plus larges. Il recouvre les opérations arithmétiques complexes (division, racine carré...), les opérations sur les matrices (multiplication de matrices 3 x 3 ou 4 x 4, transposée de matrice...) et les vecteurs, mais aussi les traitements graphiques (calcul de polygones, interpolation linéaire 2D et 3D, calcul de splines cubiques en 2D et en 3D, «clipping», convolution 3 x 3 pour le traitement d'image...).

Quatre TMS 34082 à 40 MHz peuvent être associés à un TMS 34020 sans l'apport de logique externe. La plupart des cartes compatibles Tiga ont un emplacement pour le coprocesseur TMS 34082.

A noter que Texas propose un circuit imprimé qui s'enfiche à la place du coprocesseur et comprend un TMS 34082 avec des mémoires SRAM. La plus-value due à ce coprocesseur se manifeste essentiellement dans les applications traitant de figures géométriques en 3D.

Les DSP incontournables

Les DSP se caractérisent par un cœur travaillant sur 32 bits flottants et une DMA chargée des transferts directs entre les RAM internes (usuellement deux fois 4 ko) et les mémoires externes, grâce à un très grand nombre de bus internes. Par ailleurs, le cycle d'instruction des DSP égale deux cycles d'horloge. Comme le MULAC fournit deux résultats par cycle d'instruction, les DSP débitent un résultat par cycle d'horloge au maximum. L'action simultanée de la DMA, des bus internes et des RAM internes permet de se dispenser de l'emploi de SRAM en faveur des DRAM, bien plus économiques.

Tous ces éléments se révèlent peu pertinents lorsque l'architecture impose une mémoire locale modulaire à base de SRAM, comme dans le cas du TMS 34082. La politique de prix de Texas avantage cependant le TMS 34082 qui coûte 100 dollars contre 400 à 500 dollars pour le i860 ou les DSP 32 bits.

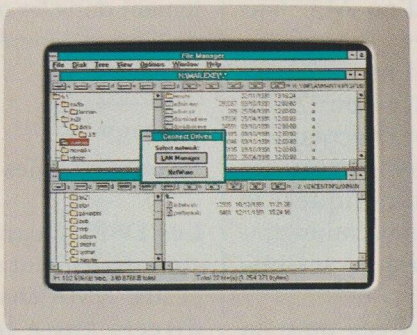
En résumé : le multiplieur et l'additionneur câblés du TMS 34082 le placent dans le haut de gamme des processeurs à virgule flottante. Le i860 peut cependant l'inquiéter sur certaines applications utilisant le MULAC et le Z-Buffer.

ROUX SEGUELA CAYZAC & GOUDARD

Nouveau, Microsoft son déploie de nouvelles portes

LAN MANAGER 2.1, ELU MEILLEUR PRODUIT DE CONNECTIVITÉ DU DERNIER COMDEX.

Le fait que Microsoft LAN Manager 2.1 ait obtenu cette distinction prouve que le jury a pu apprécier toutes les qualités d'ouverture de cette nouvelle version de LAN Manager qui vous permet d'accéder à des données de serveurs sous Netware, Appletalk, UNIX, OS/2 ou VMS.



LAN Manager 2.1 vous permet de configurer chaque poste utilisateur pour un accès transparent aux ressources du réseau, que vous souhaitiez vous connecter à un serveur LAN Manager ou à un serveur de fichiers Netware existant.

Connectivité et ouverture.

Aujourd'hui, plus d'un tiers des PC installés sont connectés en réseau. Les besoins d'interconnexion de ces réseaux vont croissant de même que la nécessité d'accéder aux ressources des mini-ordinateurs ou sites centraux en service. En intégrant en standard deux protocoles de communication, NetBEUI et TCP/IP, LAN Manager 2.1 vous permet de bâtir progressivement votre solution, du réseau départemental au réseau d'entreprise.

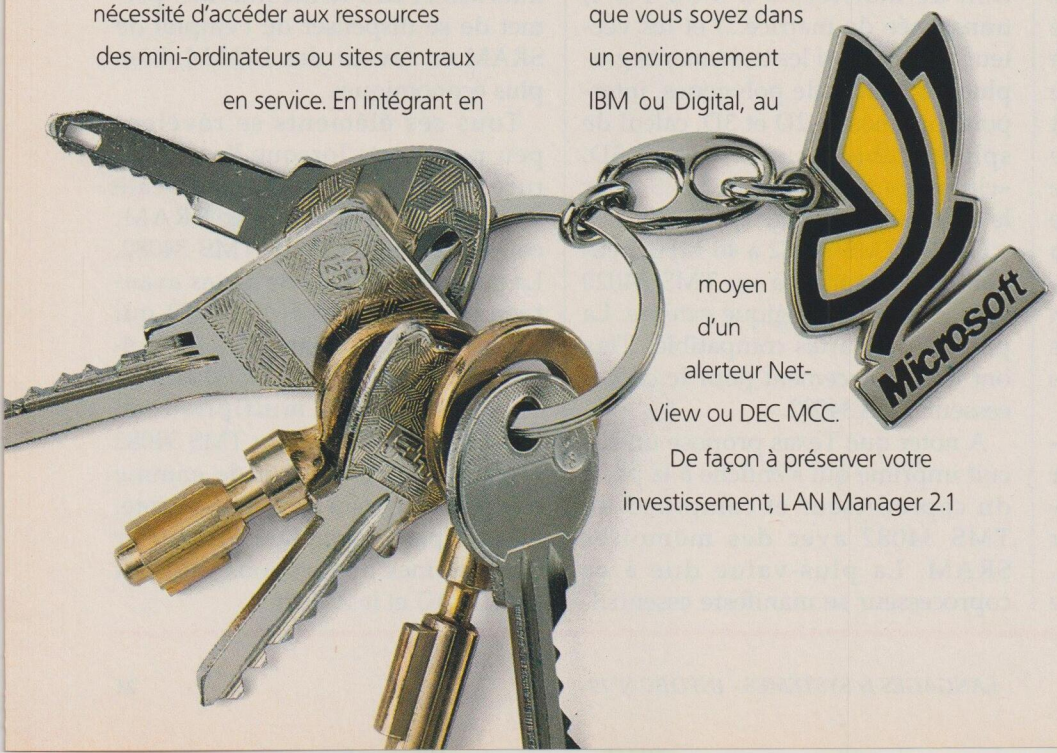
Dans ce dernier cas, au travers d'un agent SNMP, vous serez à même de superviser votre réseau, en étant compatible avec des superviseurs tels qu'Open View de Hewlett Packard ou Viewbuilder de 3Com. Bien entendu, des services analogues sont également offerts, que vous soyez dans un environnement IBM ou Digital, au moyen d'un alerteur NetView ou DEC MCC.

De façon à préserver votre investissement, LAN Manager 2.1 vous offre une solution permettant de vous connecter indifféremment sous un serveur de fichiers Netware de Novell, ou sous un serveur LAN Manager, sans encombrer inutilement la mémoire de vos postes de travail. Vos Macintosh sont désormais partie prenante de votre réseau LAN Manager. Le serveur LAN Manager se comportant comme un serveur AppleShare, vos fichiers et imprimantes PostScript deviennent immédiatement partageables entre PC et Macintosh.

Enfin, si vous souhaitez vous connecter à votre réseau depuis n'importe quelle ligne téléphonique, les services d'accès distants pour LAN Manager 2.1 vous permettront de travailler comme si vous étiez connecté localement, sans rien perdre de la facilité d'utilisation que confère un environnement tel que Microsoft Windows par exemple.

Simplicité de mise en œuvre et d'utilisation.

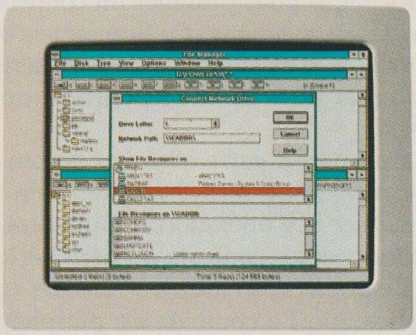
LAN Manager 2.1 est fourni en standard avec une version générique d'OS/2 optimisée et testée sur plus de deux cents plates-formes matérielles et notamment les serveurs de Compaq, d'IBM, d'Olivetti, de Hewlett Packard,



LAN Manager 2.1 : ment va ouvrir à vos micros.

de Zénith entre autres. L'administrateur du réseau trouvera les outils nécessaires à la gestion de réseaux multi-serveurs, au suivi de la charge et à l'optimisation de ceux-ci.

Du point de vue de l'utilisateur, LAN Manager 2.1 simplifie l'accès aux ressources du réseau, du fait d'une intégration étroite avec Windows 3.0 ou des versions à venir, en représentant sous forme d'icônes les serveurs et ressources associées du réseau.



L'intégration étroite de LAN Manager 2.1 et de Windows 3.0, ou ici de Windows 3.1, permet à chaque utilisateur du réseau de visualiser intuitivement les serveurs et ressources auquelles il peut accéder.

La première plate-forme Client-Serveur.

Microsoft LAN Manager constitue le premier système d'exploitation réseau client-serveur du marché comme l'attestent les quelque 150 applications recensées sous cet environnement. Du moteur de bases de données tel que Microsoft SQL Server, en passant par les serveurs de fax, ou de communication tels que DCA/Microsoft Communications Server et Atlantis/Microsoft Communications Server, vous tirez le meilleur parti de votre réseau LAN Manager sans la contrainte de dédier un serveur à chacune de ces fonctions.

LAN Manager : une solution multi plate-forme.

Simultanément à l'annonce de LAN Manager 2.1 pour OS/2, Microsoft et NCR annonçaient la version 2.0 de LAN Manager pour Unix. Cette dernière version a pour objet de permettre le déploiement de serveurs hétérogènes dans l'entreprise, que ceux-ci tournent sous OS/2 ou Unix. Plate-forme réseau indépendante du système d'exploitation, LAN Manager constitue de cette façon un ensemble de services communs à des serveurs Unix et OS/2, tant du point de vue de l'administrateur que de celui de l'utilisateur qui ne considère qu'une seule interface réseau : LAN Manager.

La création récente de l'Open LAN Manager Council, qui regroupe 22 sociétés travaillant à l'élaboration sous licence de versions compatibles de Microsoft LAN Manager, vous permet de faire le choix LAN Manager avec l'assurance d'une solution réseau homogène. Parmi ces sociétés figurent notamment : AT&T, NCR, Digital, Hewlett Packard, SCO, Olivetti, Siemens-Nixdorf Information Systems, Groupe Bull, ICL ou Zenith Data Systems.

Pour toutes informations complémentaires ou recevoir la liste des Revendeurs Spécialistes Réseaux Microsoft :
- appelez le service Clients au (1) 69 86 46 46
- ou utilisez le Minitel : 3616 code Microsoft.

Microsoft, 91957 Les Ulis Cedex et à Infomart La Défense Stand n° 276 niveau 2.

Microsoft®

LES LOGICIELS QUI DONNENT DES AILES.

Service lecteurs n° 8 (dernière page)

Toutes les marques citées sont des marques déposées.

WINDOWS

APPRENDRE WINDOWS

Explorer l'interface graphique

MARC ADLER

Grâce à la GDI, les programmeurs ont la possibilité de manipuler et d'afficher des polices (ou des fontes), des images bitmap et des métafichiers ainsi que des objets graphiques standards tels que points, lignes, cercles, ellipses, rectangles ou polygones irréguliers.

Un ensemble de fonctions GDI, nommées «fonctions BitBlt» (pour «bit-blit», abréviation de transfert de blocs de bits), est capable de déplacer, d'élargir ou de comprimer des blocs de bits entre la mémoire et l'écran.

Pour son interaction avec les périphériques d'affichage, la GDI emploie des pilotes graphiques intégrés à ce dernier. Pour permettre à Windows de supporter un nouveau périphérique, le constructeur doit écrire un pilote de périphérique spécifique.

Et comme de nombreux fabricants de périphériques souhaitent que leurs produits soient compatibles Windows, on trouve aisément les pilotes nécessaires. Ainsi, la plupart des programmes Windows peuvent fonctionner avec un large éventail de périphériques sans qu'il y ait besoin de modifier le code source.

Dans cet article, nous explorerons la GDI Windows et nous ajouterons quelques capacités graphiques à notre exemple de programme (un portefeuille d'actions) pour que chaque fenêtre MDI puisse afficher un graphe.

Le contexte de périphérique

Chaque fonction GDI utilise une structure de données appelée «contexte de périphérique» (device context). Celui-ci, agissant comme interface entre l'API Windows et le pilote du périphérique matériel, constitue la clé de l'indépendance matérielle de Windows.

Un contexte de périphérique peut être associé à n'importe quel périphérique de sortie, y compris une fenêtre, un écran entier, une imprimante, un traceur, ou même un bloc mémoire.

La plupart du temps, un contexte de périphérique se trouve associé à une fenêtre de l'écran. Ce type particulier est appelé «contexte d'affichage» (display context). Bien que les deux termes soient souvent interchangeables, nous emploierons ici le terme de «contexte d'affichage» pour désigner un contexte de périphérique associé à une fenêtre. Ce dernier se compose d'une liste d'attributs d'affichage et de caractéristiques qui déterminent la manière dont pourront opérer les fonctions GDI.

La plupart d'entre elles ont besoin d'un handle vers un contexte de périphérique. Cela signifie qu'avant d'entamer des opérations graphiques, il faut demander à Windows un contexte de périphérique. S'il y en a un de disponible, Windows renvoie un handle à votre application. Le contexte de périphérique est initialisé avec des valeurs par défaut *(voir figure 1)*.

Windows supporte quatre types de contexte d'affichage : ordinaire, de classe, privé et de fenêtre. Le plus courant d'entre eux est, bien sûr, le type ordinaire. Windows conserve un cache pour cinq contextes d'affichage et n'en attribue un que lorsqu'une application appelle une fonction GetDC ou BeginPaint.

Quand l'application a terminé ses opérations graphiques dans une fenêtre, elle doit rendre le contexte d'affichage au système à l'aide d'une fonction ReleaseDC ou EndPaint.

Le contexte d'affichage sera alors disponible pour d'autres applications. Voici un exemple typique de code gérant l'attribution d'un contexte d'affichage :

```
HDC  hDC;
case WM_COMMAND :
       if (wParam == ID_DRAWSHAPE)
       {
       hDC = GetDC(hWnd);
              .
              .
              .
       /* affichage */
              .
              .
              .
       ReleaseDC(hWnd, hDC);
       }
       break;
```

L'un des atouts majeurs de l'environnement Microsoft Windows est son moteur graphique très puissant. L'interface graphique de Windows (GDI pour Graphic Device Interface) peut fournir des graphiques indépendants du périphérique, aussi bien sur un écran monochrome que sur une imprimante laser couleur.

Le deuxième type de contexte d'affichage, le contexte de classe, se trouve partagé par toutes les fenêtres appartenant à une même classe. Pour associer un contexte de classe à une classe de fenêtre, il faut donner le style CS_CLASSDC à cette dernière au moment de son enregistrement, comme ceci :

```
WNDCLASS wc;
        .
        .
        .
wc.style = CS_CLASSDC | CS_VREDRAW | CS_HREDRAW;
RegisterClass(&wc);
```

Lorsque l'on crée le contexte d'affichage de la classe, il hérite de toutes les valeurs par défaut présentées dans la figure 1. Quand une fenêtre de cette classe à besoin de récupérer le contexte d'affichage, elle appelle l'une des fonctions GetDC ou BeginPaint, tout comme elle le ferait pour un contexte d'affichage ordinaire. La seule partie du contexte qui apparaît modifiée est la région de clipping (découpage) de la fenêtre. Les autres attributs graphiques restent inchangés.

Les fonctions ReleaseDC et EndPaint ne détruisent pas les contextes d'affichage de classe. Si vous créez une nouvelle classe de contrôle dans laquelle tous les contrôles se ressemblent, et s'il est relativement long de calculer les attributs graphiques à chaque fois qu'un contrôle doit être redessiné, un contexte d'affichage de classe permet de n'effectuer ces calculs qu'une seule fois. Chaque fenêtre de la classe possède les mêmes attributs graphiques.

Le troisième type de contexte d'affichage, le type «privé», est comparable à celui du contexte d'affichage de classe, mais il n'appartient qu'à une seule fenêtre. Comme pour le contexte de classe, les attributs graphiques sont conservés entre les appels à ReleaseDC et à GetDC : dans le cas des contextes privés, la fonction ReleaseDC s'avère en fait une opération nulle. Pour donner un contexte d'affichage privé à une fenêtre, il faut spécifier le style CS_OWNDC dans la classe de fenêtre. Le dernier type, le contexte d'affichage de fenêtre, ressemble beaucoup au type ordinaire, mais la zone de clipping du contexte d'affichage comprend les zones non clientes de la fenêtre (les bordures, le titre, les barres de défilement, les menus système, etc.). Cela permet d'afficher des dessins particuliers en dehors de la zone cliente. Pour obtenir un contexte d'affichage de fenêtre, une application doit appeler la fonction GetWindowDC au lieu de la fonction GetDC.

Les fonctions de contrôle de périphérique

Plusieurs fonctions de Windows concernent la création, la destruction et la sauvegarde des contextes de périphérique. Windows conserve une pile interne de contexte qui peut être employée pour stocker et récupé-

FIGURE 1 : ATTRIBUT PAR DÉFAUT DES CONTEXTES DE PÉRIPHÉRIQUE

ATTRIBUT	DEFAUT
Couleur d'arrière-plan	Blanc
Mode d'arrière-plan	OPAQUE
Bitmap	Pas de valeur par défaut
Pinceau	WHITE_BRUSH
Origine du pinceau	(0, 0)
Région de clipping (limites)	Surface d'affichage
Palette de couleurs	DEFAULT_PALETTE
Position courante du crayon	(0, 0)
Mode d'affichage	R2_COPYPEN
Police	SYSTEM_FONT
Espace entre les caractères	0
Mapping mode	MM_TEXT
Crayon	BLACK_PEN
Mode de remplissage (polygones)	ALTERNATE
Mode de stretching	BLACKONWHITE
Couleur de texte	Noir
Extent du viewport	(1, 1)
Origine du viewport	(0, 0)
Extent de la fenêtre	(1, 1)
Origine de la fenêtre	(0, 0)

rer les informations relatives aux attributs d'un contexte de périphérique. La fonction SaveDC sauvegarde sur le sommet de la pile les attributs contenus dans le contexte qui lui est spécifié; la fonction RestoreDC permet de récupérer n'importe lequel des contextes sauvegardés dans cette pile. Voici un exemple de code :

```
SaveDC(hDC);
        .
        .
        .
RestoreDC(hDC, -1);
```

L'argument -1 dans l'appel à RestoreDC indique que le dernier contexte sauvegardé doit être extrait de la pile de contexte. La fonction CreateDC crée un nouveau contexte de périphérique. Elle est habituellement employée lorsque l'application doit accéder à l'imprimante ou à un autre périphérique matériel. Elle effectue un lien entre l'application, le pilote de périphérique et le périphérique matériel.

Lorsque l'on n'utilise plus un contexte de périphérique alloué par la fonction CreateDC, il faut appeler DeleteDC afin de le détruire. Cette fonction libère la mémoire occupée par le contexte de périphérique. Si Windows détecte qu'il s'agit du dernier contexte de périphérique associé à un périphérique donné, toutes les ressources employées par ce dernier sont libérées. Voici un exemple typique d'utilisation de CreateDC et de DeleteDC :

```
hPrinterDC = CreateDC("HPPCL", "PCL / PLaserJet",
                      "LPT1:", (LPDEVMODE) NULL);
        .
        .
        .
DeleteDC(hPrinterDC);
```

Les arguments passés à CreateDC sont les noms du pilote et du périphérique de sortie, le fichier ou le port de sortie utilisé par le périphérique et, enfin, un pointer sur une structure DEVMODE. Cette dernière contient l'information d'initialisation du périphérique. Si l'on donne NULL comme dernier argument, c'est Windows qui fournit une information d'initialisation par défaut à ce périphérique. Si l'on désire obtenir une information concernant un périphérique sans créer un contexte de périphérique, on peut employer un «contexte d'information».

Celui-ci est comparable à un contexte de périphérique, à la différence qu'il ne peut pas être utilisé pour envoyer des données vers le périphérique. Pour obtenir un contexte d'information relatif à un périphérique, on appelle la fonction CreateIC qui possède les mêmes paramètres que CreateDC et qui renvoie un HDC. Quand on a obtenu un handle sur un contexte d'information, on peut alors appeler des fonctions telles que GetDeviceCaps pour récupérer une information concernant le périphérique.

Lorsque le contexte d'information n'est plus utile, il faut appeler DeleteDC, comme s'il s'agissait d'un contexte de périphérique normal. Voici un exemple d'utilisation d'un contexte d'information :

```
/*
Obtenir le nombre de couleurs supportées par
l'imprimante
*/
HDC hIC;
int nPlanes;
hIC = CreateIC("HPPCL", "PCL / HPLaserJet", "LPT1:",
                                (LPDEVMODE) NULL);
nColorrs = GetDeviceCaps(hIC, NUMCOLORS);
DeleteDC(hIC);
```

La dernière fonction importante, en ce qui concerne les contextes de périphérique, est CreateCompatibleDC. Elle génère un contexte de périphérique mémoire qui est un bloc mémoire que l'application peut traiter comme une surface d'affichage. Nous examinerons cette fonction quand nous étudierons la manière dont Windows se sert des images bitmap.

Affichage graphique

Windows s'efforce de tirer avantage des caractéristiques d'un périphérique de sortie afin de produire des objets graphiques de haut niveau. Par exemple, certains adaptateurs graphiques disposent de composants particuliers pour afficher très rapidement les images BitBlt et d'autres possèdent des caractéristiques permettant d'afficher rapidement des objets graphiques, tels que les cercles. Si ces spécificités sont gérées par le pilote du périphérique, Windows tente de les utiliser afin d'afficher les objets graphiques le plus rapidement possible. La fonction graphique Windows la plus «primitive» est SetPixel, qui dessine simplement un pixel sur le périphérique de sortie à un emplacement déterminé et dans une couleur spécifiée. Voici l'appel employé pour dessiner un pixel noir au croisement de la ligne 20 et de la colonne 10 :

```
SetPixel(hDC, 10, 20, RGB(0, 0, 0));
```

On pourrait employer une suite d'appels à SetPixel pour dessiner n'importe quelle sorte d'objet graphique sur le périphérique. Heureusement, Windows possède des fonctions graphiques beaucoup plus avancées autorisant l'affichage des objets graphiques de haut niveau.

Les attributs d'affichage associés au contexte d'affichage offrent également la possibilité de remplir ces objets avec une couleur ou un motif, de choisir les styles de crayon pour tracer les contours et de combiner les bits composant l'objet avec ceux qui se trouvent déjà sur l'écran. Nous reviendrons en détail sur ce sujet.

Les lignes

Pour dessiner un simple segment de ligne, il faut appeler deux fonctions. La fonction MoveTo place le crayon courant à un emplacement précisé de la fenêtre et la fonction LineTo dessine une ligne à partir de celui-ci jusqu'à une nouvelle position. A titre d'exemple, pour dessiner une ligne allant des coordonnées (10, 20) à (30, 50), il nous faut ces deux instructions :

```
MoveTo(hDC, 10, 20);
LineTo(hDC, 30, 50);
```

Une fois la ligne dessinée, la fonction LineTo déplace la position courante du crayon à l'extrémité de la ligne. Si la suivante part de ce nouvel emplacement, il n'est plus nécessaire de rappeler MoveTo. Voici comment afficher un rectangle sur l'écran :

```
MoveTo(hDC, 10, 20);
LineTo(hDC, 100, 20);
LineTo(hDC, 100, 50);
LineTo(hDC, 10, 50);
LineTo(hDC, 10, 20);
```

Les fonctions MoveTo et LineTo sont les seules fonctions GDI qui modifient la position courante du crayon.

Les crayons

Plusieurs attributs de dessin déterminent l'aspect d'une ligne. Le plus important est le crayon (appelé parfois «stylo»). Il constitue un objet graphique qui définit la manière dont sont tracés les lignes et les contours (les crayons ne remplissent pas les zones avec une couleur ou un motif; un autre outil est employé pour cela : le pinceau).
Un crayon possède trois caractéristiques :
- le style de la ligne,
- la largeur (l'épaisseur) de la ligne,
- la couleur.

Le crayon par défaut dessine une ligne noire unie ayant une épaisseur équivalente à 1 unité logique. Il est possible de créer un nouveau crayon à l'aide d'une des fonctions CreatePen ou CreatePenIndirect. Lorsqu'un nouveau crayon a été conçu — et que son handle a été renvoyé à l'application —, il faut l'associer à un contexte d'affichage avec la fonction SelectObject. Voici un exemple de code pour utiliser un crayon traçant une ligne bleue discontinue (en tirets) :

```
HPEN hPen;
hPen = CreatePen(PS_DASH, 1, RGB(0x00, 0x00,
0xFF)); hOldPen = SelectObject(hDC, hPen);
MoveTo(hDC, 10, 20); LineTo(hDC, 30, 50);
SelectObject(hDC, hOldPen); DeleteObject(hPen);
```

Les styles de crayon sont définis dans le fichier WINDOWS.H et commencent par le préfixe PS_. On emploie le style PS_SOLID pour un crayon dessinant en continu, le style PS_DOT pour un crayon usant du pointillé. Malheureusement, Windows ne permet pas à l'utilisateur de créer ses propres styles.

Dans l'exemple de code précédent, la fonction CreatePen élabore un nouveau crayon et renvoie un handle sur ce crayon. Le type de données HPEN définit un handle sur un crayon et doit toujours être employé pour la gestion des crayons (en utilisant typedef, on protège normalement le code source de toute modification qui pourrait affecter les données internes de Windows).

Lorsqu'un nouveau crayon est «fabriqué», ses propriétés doivent être rattachées au contexte d'affichage grâce à la fonction SelectObject. Celle-ci renvoie un handle sur le précédent objet de ce type associé au contexte d'affichage spécifié. Il s'agit dans ce cas d'un handle sur le crayon antérieur.

Le nouveau style de crayon est employé par la fonction LineTo. Quand l'objet que nous avons créé n'est plus utile, il faut sélectionner de nouveau l'objet précédent dans le contexte d'affichage, puis détruire l'objet créé.

Comme il n'existe pas de fonction DeletePen, nous utilisons la fonction générique DeleteObject pour supprimer les objets d'affichage standard tels que les crayons, les pinceaux, les polices et les images bitmap.

Les figures

Windows possède un certain nombre de fonctions permettant de dessiner des figures graphiques : rectangles, ellipses, polygones ou groupes de polygones. Le contour de chaque objet est tracé avec le style de crayon courant, et son contenu se trouve rempli avec le style de pinceau courant.

La fonction Rectangle reçoit cinq arguments : un handle sur un contexte d'affichage et quatre coordonnées spécifiant le coin supérieur gauche et le coin inférieur droit du rectangle *(voir figure 2)*.

```
Rectangle(hDC, 10, 20, 240, 150);
```

Un polygone est constitué d'une suite de segments reliés par leurs extrémités. La fonction Polygon utilise un handle sur le contexte d'affichage, un tableau de points et une valeur indiquant le nombre de points contenus dans le tableau.

Pour afficher un triangle, par exemple, nous pouvons écrire *(voir figure 3)*:

```
POINT aptTriangle[]
                = { { 10, 20 }, { 30, 100 }, { 0, 100 } };
Polygon(hDC, (LPPOINT) aptTriangle, 3);
```

Bien que l'extrémité du dernier segment n'ait pas été indiquée, la fontion Polygon referme automatiquement le polygone sur le premier point du tableau. Les objets elliptiques supportés par la GDI Windows comprennent les ellipses, les «parts de camembert» et les cordes.

La fonction Ellipse utilise les mêmes arguments que la fonction Rectangle : on spécifie les coordonnées d'un rectangle contenant l'ellipse à afficher *(voir figure 4)*.

```
Ellipse(hDC, 10, 20, 240, 150);
```

Un cercle est une ellipse dont les deux axes sont de même longueur. Le rectangle englobant devient alors un carré. Voici comment dessiner un cercle *(voir figure 5)*:

```
Ellipse(hDC, 10, 20, 110, 120);
```

En fait, ce n'est pas vrai pour l'ensemble des cas. Cette méthode fonctionne dans tous les modes d'affichage à l'exception de MM_TEXT et MM_ANISOTROPIC, car les proportions des périphériques d'affichage peuvent varier.

Si nous écrivons un programme pour afficher un rectangle de 10 pixels de haut sur 10 pixels de large et si nous posons une règle sur l'écran pour mesurer ce rectangle, nous pourrons remarquer que ses côtés ne sont pas égaux.

Dans les deux modes spécifiés plus haut, il faudra appeler GetDeviceCaps pour obtenir le facteur de correction des proportions du périphérique, puis redéfinir les limites du rectangle englobant en fonction de ce facteur afin d'obtenir un cercle bien rond.

FIGURE 2 : AFFICHER UN RECTANGLE

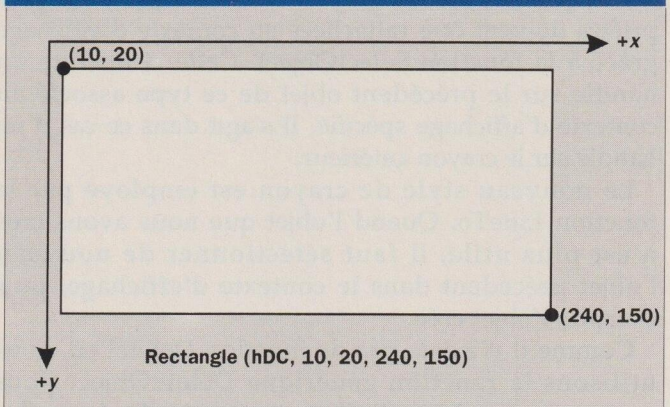


FIGURE 3 : AFFICHER UN TRIANGLE

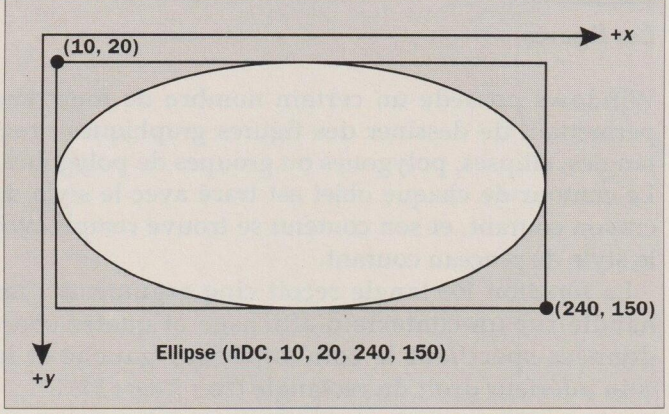


FIGURE 4 : AFFICHER UNE ELLIPSE

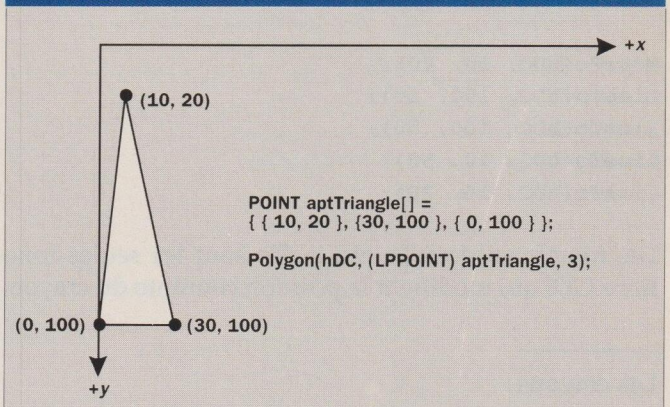
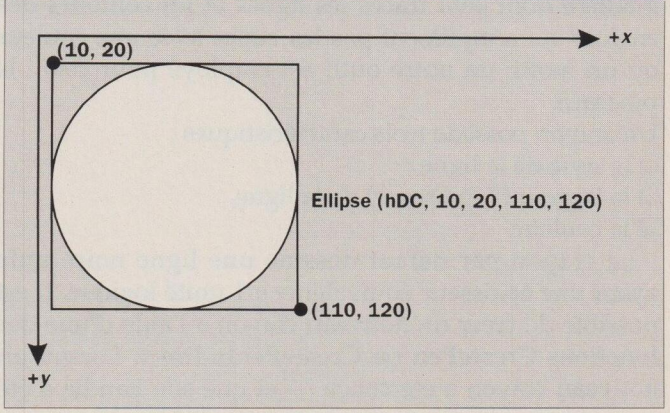


FIGURE 5 : AFFICHER UN CERCLE AVEC LA FONCTION ELLIPSE



Les pinceaux et modes d'affichage et d'arrière-plan

Un pinceau est un objet d'affichage composé d'une couleur unie, d'un motif hachuré ou une image bitmap indépendante du périphérique (image DIB, pour Device Independent Bitmap). Grâce aux pinceaux bitmap, il est possible de peindre dans une fenêtre avec des motifs personnalisés. Plusieurs fonctions permettent de créer des pinceaux. CreateSolidBrush renvoie un pinceau capable de remplir un objet avec une couleur unie. Des pinceaux unis prédéfinis peuvent être obtenus à l'aide de la fonction GetStockObject. Voici deux manières de générer un pinceau noir uni :

```
hBrush = GetStockObject(BLACK_BRUSH);
ou
hBrush = CreateSolidBrush(RGB( 0x00, 0x00, 0x00));
```

Si l'on ne souhaite pas que l'intérieur d'un objet soit peint, on peut employer le pinceau NULL :

```
hBrush = GetStockObject(NULL_BRUSH);
hOldBrush = SelectObject(hDC, hBrush);
```

La fonction CreateHatchBrush permet de créer un pinceau hachuré. Le motif est composé de fines lignes espacées régulièrement. Windows propose, par exemple, un motif hachuré constitué de lignes inclinées à 45° vers la gauche. Six différents motifs hachurés sont disponibles *(voir figure 6)*.

```
/* Créer un pinceau vert avec un motif croisé */
hBrush =  CreateHatchBrush(HS_CROSS,
                    RGB( 0x00, 0xFF, 0x00);
```

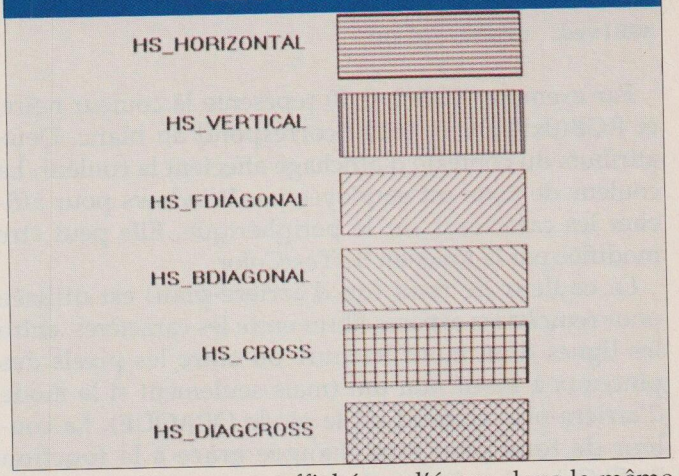
Un motif personnalisé peut être créé pour un pinceau à l'aide de la fonction CreatePatternBrush, à laquelle il faut passer le handle d'une image bitmap dont la taille est au moins égale à 8 x 8 pixels. Le pinceau répétera le motif bitmap jusqu'à ce que l'objet graphique se trouve complètement rempli. On peut encore utiliser une image DIB comme motif en appelant la fonction CreateDIBPatternBrush.

Un attribut appelé «mode d'affichage» (ou mode de dessin : drawing mode) détermine la manière dont les bits des pinceaux et des crayons sont combinés avec ceux déjà présents sur l'écran. La fonction SetROP2 (SetRasterOp) contrôle le mode d'affichage associé à un contexte d'affichage. Il existe 16 modes d'affichage, donnant toutes les combinaisons logiques du pinceau ou du crayon avec l'écran.

La liste de ces modes apparaît dans le fichier WINDOWS.H et est présentée dans la figure 7 de la page suivante; ils commencent tous par le préfixe R2_.

Le mode par défaut est R2_COPYPEN qui signifie

FIGURE 6 : MOTIFS PRÉDÉFINIS DES PINCEAUX



que le pixel se trouve affiché sur l'écran dans la même couleur que le pixel correspondant du pinceau ou du crayon.

Le mode R2_NOT inverse simplement la couleur du pixel déjà affiché sur l'écran, sans tenir compte du pixel correspondant du pinceau ou du crayon.

Pour donner le style R2_NOT au mode d'affichage courant, on effectue l'appel suivant :

```
SetROP2(hDC, R2_NOT);
```

L'attribut du mode d'arrière-plan contrôle l'état des espaces entre les caractères affichés sur l'écran, entre les lignes d'un motif hachuré ou les tirets d'un crayon (s'il n'est pas uni).

Il existe deux modes :

- Le mode OPAQUE, mode par défaut, indique à Windows qu'il faut remplir les espaces avec la couleur d'arrière-plan courante.
- Le mode TRANSPARENT laisse intacts les pixels contenus dans les espaces. Le mode d'arrière-plan est défini par un appel de ce genre :

```
SetBkMode(hDC, TRANSPARENT);
```

Les couleurs

Windows emploie un format de couleur sur 24 bits appelé généralement «RGB» (pour Red, Green, Blue : rouge, vert, bleu). Avec ce format, toutes les couleurs sont composées d'une combinaison de différentes intensités des trois couleurs primaires de la lumière : le rouge, le vert et le bleu.

Sous Windows, l'intensité de chaque primaire varie entre 0 et 255. Une couleur est exprimée par les 24 bits de poids faible d'un double mot de 32 bits. En partant de celui qui a le poids le plus faible, les trois octets inférieurs contiennent respectivement l'intensité du bleu,

celle du vert et celle du rouge. Une macro définie dans WINDOWS.H permet d'élaborer une couleur :

```
RGB(red, green, blue)
```

Par exemple, RGB(0, 0, 0) repésente la couleur noire, et RGB(0xFF,0xFF, 0xFF) correspond au blanc. Deux attributs du contexte d'affichage affectent la couleur. La couleur de texte est employée par Windows pour afficher les caractères sur le périphérique. Elle peut être modifiée par la fonction SetTextColor.

La couleur de fond (ou d'arrière-plan) est utilisée pour remplir les espaces libres entre les caractères, entre les lignes d'un motif hachuré ou entre les pixels des pinceaux à motif non uni (mais seulement si le mode d'arrière-plan courant est le mode OPAQUE). La couleur de fond peut être changée grâce à la fonction SetBkColor. Les instructions suivantes définissent une couleur de fond rouge et une couleur de texte jaune :

```
SetBkColor(hDC, RGB(0xFF, 0, 0);
SetTextColor(hDC, RGB(0, 0xFF, 0xFF));
```

Les «mapping modes»

Les «mapping modes» représentent l'une des caractéristiques les plus souples de Windows, mais aussi l'une des plus complexes (il s'agit de modes définissant le système de coordonnées, mais nous conserverons le terme «mapping mode» employé couramment par les programmeurs).

Un mapping mode définit l'orientation des axes x et y d'une fenêtre ainsi que l'unité logique de mesure employée par les fonctions GDI.

Une unité logique représente une subdivision abstraite d'une fenêtre. Jusqu'à présent, nous n'avons simplement considéré qu'une fenêtre était composée de pixels. En fait, nous étions tout le temps dans le mapping mode MM_TEXT. Dans celui-ci, une unité périphérique (un pixel) correspond à une unité logique. Avec l'instruction suivante ,

```
SetPixel(hDC, 20, 10, RGB(0, 0, 0));
```

un pixel est affiché à 20 pixels du bord gauche et à 10 pixels du sommet de la fenêtre. Il faut établir une distinction entre unité logique et unité périphérique. Une unité périphérique de l'écran équivaut à un pixel. Si l'adaptateur vidéo possède une résolution de 640 x 480 pixels, cela signifie que l'écran a une largeur de 640 unités périphériques et une hauteur de 480 unités périphériques.

Une fenêtre de 100 x 200 a une largeur de 100 pixels et une hauteur de 200 pixels. La valeur des unités périphériques est déterminée au cours d'une session Windows.

Mais comment faire si nous écrivons un programme qui doit mesurer une fenêtre d'une autre manière? Des programmes de mise en pages, comme Aldus PageMaker, doivent mesurer une fenêtre en pouces ou en centimètres plutôt qu'en pixels.

Dans un système graphique qui ne supporte pas les mapping modes, il faut calculer la hauteur et la largeur de l'écran, le facteur de correction des proportions et la résolution courante, puis déterminer le nombre de pixels correspondant à un pouce ou à un centimètre.

Dans ce premier cas, c'est le pouce qui constitue l'unité logique. Nous divisons (abstraitement) la fenêtre en pouces.

Mais une résolution de 1 pouce serait probablement trop grossière pour une application. On emploie plus couramment des centièmes ou des millièmes de pouce.

FIGURE 7 : CODES DES MODES D'AFFICHAGE POUR SETROP2

VALEUR	Couleur du pixel
R2_BLACK	Toujours noir
R2_WHITE	Toujours blanc
R2_NOP	Inchangé
R2_NOT	Inverse de la couleur affichage
R2_COPYPEN	Couleur du crayon
R2_NOTCOPYPEN	Inverse de la couleur du crayon
R2_MERGEPENNOT	Combinaison de la couleur du crayon et de l'inverse de la couleur d'affichage
R2_MASKPENNOT	Combinaison des couleurs communes au crayon et à l'inverse de l'affichage
R2_MERGENOTPEN	Combinaison de la couleur d'affichage et de l'inverse de la couleur du crayon
R2_MASKNOTPEN	Combinaison des couleurs communes à l'affichage et à l'inverse du crayon
R2_MERGEPEN	Combinaison de la couleur du crayon et de la couleur d'affichage
R2_NOTMERGEPEN	Inverse de la couleur de R2_MERGEPEN
R2_MASKPEN	Combinaison des couleurs communes au crayon et à l'affichage
R2_NOTMASKPEN	Inverse de la couleur de R2_MASKPEN
R2_XORPEN	Combinaison des couleurs du crayon et de l'affichage, mais pas des couleurs communes
R2_NOTXORPEN	Inverse de la couleur de R2_XORPEN

Il ne suffit pas de convertir les unités logiques en unités périphériques, il faut encore changer le système de coordonnées.

Avec le mapping mode MM_TEXT, les valeurs de x augmentent vers la droite (axe horizontal orienté à droite) et les valeurs de y augmentent vers le bas (axe vertical orienté vers le bas). Ainsi, le point (10, 20) apparaît plus bas et plus à droite que le point (5, 10). Mais, dans un système de coordonnées cartésien, les valeurs de y augmentent vers le haut, à partir de l'origine.

De nombreuses applications doivent utiliser ce système pour afficher des graphes professionnels. Heureusement, Windows propose huit mapping modes différents, chacun supportant une unité logique spécifique.

Le mapping mode est défini par un appel à la fonction SetMapMode, ce qui affecte tous les appels ultérieurs à des fonctions GDI. Les coordonnées passées aux fonctions GDI doivent être exprimées en unités logiques et non en pixels ou en unités périphériques.

On ne peut pas établir une équivalence exacte entre des pixels et des coordonnées de fenêtre — cela ne marche qu'avec le mode standard MM_TEXT, mais il faut penser en termes d'unités logiques si l'on veut explorer la GDI. Pour un programme de mise en pages, on se sert des mapping modes MM_LOENGLISH ou MM_HIENGLISH.

Dans le premier, une unité logique correspond à 1/100 de pouce et l'on utilise des coordonnées cartésiennes. Pour afficher un pixel dans une fenêtre, à 1 pouce vers la droite et à 1 pouce au-dessous de l'origine, nous écrirons l'instruction suivante :

```
SetMapMode(hDC, MM_LOENGLISH); SetPixel, hDC,
                    100, -100, RGB(0, 0, 0));
```

Dans cet appel à SetPixel, il peut paraître surprenant de spécifier la valeur -100 pour la coordonnée y. Bien que le mapping mode MM_LOENGLISH utilise un système de coordonnées cartésien, l'origine de la fenêtre reste toujours le coin supérieur gauche de cette dernière! Par conséquent, une coordonnée y de -100 en mode MM_LOENGLISH représente une position située à 1 pouce au-dessous de l'origine.

Intuitivement, pour beaucoup de gens, il semblerait plus évident de déplacer l'origine vers le coin inférieur gauche de la fenêtre. Windows a prévu cela en incluant cette information dans un contexte de périphérique.

Deux concepts se trouvent étroitement liés aux mapping modes, les fenêtres et les «viewports». Une fenêtre est une zone mesurée en coordonnées logiques; un viewport constitue une zone mesurée en coordonnées périphériques. Comme les fenêtres, les viewports ont une origine et des caractéristiques de proportion (extents). L'origine définit l'intersection des axes x et y, où se situe le point (0,0).

On considère l'origine d'un viewport comme le point logique où se place le point (0, 0) d'une fenêtre. Les extents de la fenêtre et du viewport définissent des proportions horizontale et verticale qui sont employées par Windows pour établir une correspondance entre les dimensions de la fenêtre et celles du viewport. Pour simplifier, nous pouvons résumer ce que nous venons de dire par ces deux formules :

```
xViewport =   (xWindow - xWindowOrigin) *
              (xViewportExtent / xWindowExtent) +
              xViewportOrigin
yViewport =   (yWindow - yWindowOrigin) *
              (yViewportExtent / yWindowExtent) +
              yViewportOrigin
```

Dans ces formules, les variables xWindow et yWindow sont les coordonnées logiques, tandis que xViewport et yViewport constituent les coordonnées périphériques. On détermine la distance entre une coordonnée de fenêtre et l'origine de cette dernière, on multiplie cette distance par un facteur d'échelle et on ajoute la nouvelle distance à l'origine du viewport.

Bien entendu, les fonctions de la GDI Windows permettent de définir et d'obtenir les origines et les extents d'une fenêtre ou d'un viewport. Mais, dans plusieurs mapping modes, les extents ne peuvent pas être modifiés.

Il s'agit du mode MM_TEXT et des cinq modes métriques : MM_LOENGLISH, MM_HIENGLISH, MM_TWIPS, MM_LOMETRIC et MM_HIMETRIC. Dans ces modes, c'est Windows qui calcule les facteurs d'échelle en fonction de la taille de l'écran et du facteur de correction des proportions.

Ce qui explique pourquoi il n'est pas possible de changer le facteur d'échelle.

Les fonctions SetWindowOrg et SetViewportOrg modifient respectivement l'origine de la fenêtre et du viewport. Pour que le point logique (0, 0) soit situé dans le coin inférieur gauche d'une fenêtre en mode MM_LOENGLISH, on peut utiliser ces instructions :

```
SetMapMode(hDC, MM_LOENGLISH);
GetClientRect(hWnd, (LPRECT) &rClient);
SetViewportOrg(hDC, 0, rClient.bottom);
```

Le mapping mode MM_ISOTROPIC permet de conserver la forme d'un objet avec n'importe quel facteur de correction des proportions.

Windows ajuste les extents horizontaux et verticaux pour que les unités logiques verticales puissent avoir la même taille que les unités logiques horizontales.

Cela signifie qu'une ellipse ayant une hauteur et une

largeur égales sera toujours affichée comme un cercle, même si l'utilisateur modifie les dimensions de la fenêtre.

Un bon exemple est représenté par l'accessoire Horloge de Windows. La première fois que l'on exécute ce programme, l'horloge a la forme d'un cercle parfait.

Si vous changez la taille de la fenêtre, l'horloge reste bien ronde et ne se déforme pas pour s'inscrire dans la fenêtre. C'est l'inverse avec le mapping mode MM_ANISOTROPIC.

Il n'ajuste pas automatiquement les extents de la fenêtre et du viewport lorsqu'on modifie les dimensions de la fenêtre. Les objets contenus dans celle-ci sont alors étirés ou comprimés.

Si le programme Horloge utilisait le mode MM_ANISOTROPIC et si l'on donnait à la fenêtre la forme d'un rectangle très allongé, l'horloge serait étirée en conséquence pour apparaître comme une ellipse très allongée.

Si l'on se trouve dans un des modes MM_ISOTROPIC ou MM_ANISOTROPIC, on peut définir à la fois les extents de la fenêtre et ceux du viewport, respectivement à l'aide des fonctions SetWindowExt et SetViewportExt.

Ces deux fonctions sont employées par notre application d'exemples pour afficher les points sur les graphes des actions. Si l'origine du système de coordonnées est placée à un coin de la fenêtre, l'extent du viewport prend généralement pour valeur la taille de cette dernière.

Les extents de la fenêtre prennent pour valeur le nombre d'unités logiques dans les deux directions. Dans le graphe d'une action, par exemple, l'axe y va de 0 à la valeur maximale de l'action. L'axe x est divisé par le nombre maximal de jours à afficher.

```
SetMapMode(hDC, MM_ISOTROPIC);
GetCLientRect(hWnd, (LPRECT) &rClient);
SetWindowExt(hDC, Stock.maxTickDays, Stock.maxPrice);
        SetViewportExt(hDC,
        rClient.right - rClient.left,
        rClient.top - rClient.bottom);
SetViewportOrg(hDC, 0, rClient.bottom);
```

Avec ces instructions, il est facile d'imaginer que les axes de la fenêtre sont divisés en jours et en dollars. Si une action s'échangeait à 40 dollars le dixième jour, nous afficherons le point avec :

```
setPixel(hDC, 10, 40);
```

Une application doit parfois convertir une unité logique en unité périphérique, ou vice versa. Windows offre deux fonctions de conversion : DPtoLP et LPtoDP. DP signifie Device Point (point périphérique) et LP correspond à Logical Point (point logique).

Ces deux fonctions prennent comme arguments un handle sur un contexte d'affichage, un tableau de points et un entier spécifiant le nombre de points dans le tableau. Les points du tableau sont convertis, d'un système dans l'autre. Voici un exemple pour déterminer si le point logique (0, 0) apparaît sur le périphérique de sortie :

```
POINT pt;
pt.x = pt.y = 0;
LPtoDP(hDC, (LPPOINT) &pt, 1);
```

Il faut du temps pour se familiariser avec les mapping modes, les fenêtres, les viewports, les coordonnées logiques et périphériques. Il ne faut pas s'inquiéter si tout cela ne paraît pas immédiatement évident.

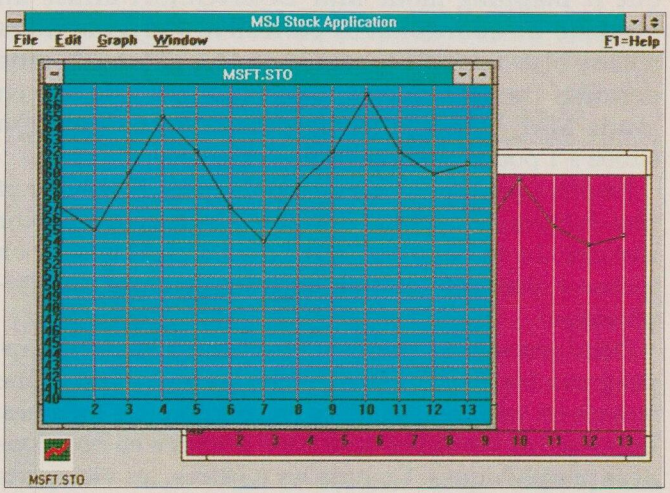
Revenons à notre application

Maintenant que nous avons étudié une partie de la GDI Windows, nous pouvons ajouter les graphes à notre application de gestion d'un portefeuille d'actions *(voir figure 8)*.

La fonction GraphWndPaint, fournie par le fichier source GRAPH.C *(voir listing en fin d'article)*, dessine le contenu de la zone cliente d'une fenêtre fille MDI. Bien sûr, le composant principal de la zone cliente est le graphe de l'action. GraphWndPaint est appelée à chaque fois que Windows envoie un message WM_PAINT à la WinProc du document MDI.

Le message WM_PAINT informe une fenêtre que tout ou partie de sa zone cliente doit être redessiné. Il peut être généré par Windows lui-même quand il remarque qu'une portion de fenêtre préalablement cachée est devenue visible.

Par exemple, si la fenêtre se trouve sous une pile (une



Notre application affiche enfin des graphes!

cascade) de fenêtres overlaped et que l'utilisateur clique dans la barre de titre, la fenêtre en question doit être amenée au-dessus de la cascade. Pour informer l'application que la zone cliente de la fenêtre doit être redessinée, Windows envoie le message WM_PAINT à la WinProc.

Une fenêtre nécessite également un nouveau dessin lorsqu'elle a été déplacée, élargie ou rétrécie. L'application elle-même peut demander le rafraîchissement (la mise à jour) d'une partie ou de la totalité de la zone cliente d'une fenêtre.

Pour cela, elle appelle une des fonctions InvalidateRect ou InvalidateRegion qui ajoutent un ou plusieurs rectangles à la zone de mise à jour d'une fenêtre.

Chaque fenêtre contient une région de mise à jour, constituée par une liste d'une ou de quelques zones rectangulaires dans la zone cliente à rafraîchir.

Quand l'application invalide un rectangle ou une région d'une fenêtre en appelant InvalidateRect ou InvalidateRegion, Windows ajoute les rectangles à la région de mise à jour de cette fenêtre puis place un message WM_PAINT (pour cette fenêtre) à la fin de la file d'attente des messages.

Les messages d'affichage ont un niveau de priorité très bas, et les messages de haute priorité sont placés dans la file devant WM_PAINT.

Si la file contient deux messages WM_PAINT destinés à la même fenêtre, ils sont combinés. En d'autres termes, la région de mise à jour référencée par le second message WM_PAINT se trouve combinée avec la zone de mise à jour indiquée dans le premier message.

Une application peut forcer une mise à jour immédiate en appelant UpdateWindow. Cette fonction envoie le message WM_PAINT à la WinProc de la fenêtre spécifiée et supprime de la file d'attente tous les messages WM_PAINT destinés à cette fenêtre. Pour forcer la mise à jour d'une fenêtre entière, vous pouvez placer les instructions suivantes dans votre application :

```
InvalidateRect(hWnd, (LPRECT) NULL, TRUE);
UpdateWindow(hWnd);
```

Le deuxième argument d'InvalidateRect précise le rectangle qui doit être placé dans la région de mise à jour (si la valeur NULL est passée, toute la zone cliente se situe dans la région de mise à jour).

Le troisième argument indique si la zone cliente doit être effacée. L'appel à UpdateWindow provoque l'envoi du message WM_PAINT vers la fenêtre spécifiée. Un aspect intéressant (certains diront énervant) de Windows est que la responsabilité de l'affichage de la zone cliente d'une fenêtre incombe à l'application et pas à Windows.

La plupart des systèmes alphanumériques conservent le contenu d'une fenêtre ou d'un écran dans un tampon mémoire. Cela fonctionne bien pour les modes texte, car il suffit de 4 000 octets pour sauvegarder un écran de 80 x 25 caractères.

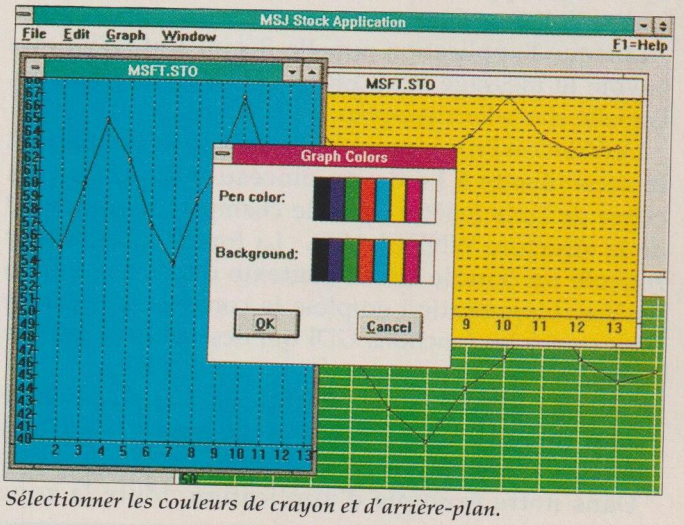
Mais il y aurait besoin d'une énorme quantité de mémoire pour sauvegarder le contenu d'une fenêtre graphique, car il serait nécessaire de préserver toutes les informations concernant les pixels. Avec une résolution vidéo de 640 x 350, il faudrait sauvegarder 224 000 pixels.

Le nombre d'octets dépend de la quantité de bits indispensables pour représenter un pixel. Il faut un octet par pixel avec un système Super VGA offrant 256 couleurs. Windows économise la mémoire en demandant aux applications de dessiner elles-mêmes le contenu de leurs zones clientes.

Il y a un choix à faire entre les performances et l'économie de la mémoire. Une WinProc doit réagir à la réception d'un message WM_PAINT en appelant la fonction BeginPaint avant tout affichage dans la fenêtre. A la fin du traitement de WM_PAINT, elle doit appeler la fonction EndPaint. Voici la structure de traitement d'un message WM_PAINT :

```
HDC hDC;
PAINTSTRUCT ps;
case WM_PAINT :
        hDC = BeginPaint(hWnd, (LPPAINTSTRUCT) &ps);
        .
        .
        .
        EndPaint(hWnd, (LPPAINTSTRUCT) &ps);
```

La fonction BeginPaint remplit la structure PAINTSTRUCT avec les informations relatives à la mise à jour. PAINTSTRUCT contient trois éléments : un handle sur



Sélectionner les couleurs de crayon et d'arrière-plan.

FIGURE 11 : AFFICHER LES INDICATIONS DES AXES ET LES REPÈRES

```
/*
Afficher les repères de l'axe des x
*/
for (i = 1; i < lpStockInfo->StockFile.nTicks; i++)
{
        int x = i * xPixelsPerDate + yAxisStart;
        int iTextLen;
        MoveTo(hDC, x, xAxisStart-2);
        LineTo(hDC, x, xAxisStart+2);
        sprintf(szBuf, "%d", i+1);
        iTextLen = strlen(szBuf) * tm.tmAveCharWidth;
        /*
        Centrer le texte sur le repère
        */
        TextOut(hDC, x-(iTextLen>>1),
                xAxisStart-2, szBuf,strlen(szBuf));
        }
/*
Afficher les repères de l'axe des y
*/
for (i = rangeLow; i < rangeHigh; i++)
{
    int y = (i-rangeLow) * yGraph/(rangeHigh-rangeLow) + xAxisStart;
    MoveTo(hDC, yAxisStart-4, y);
    LineTo(hDC, yAxisStart+4, y);    sprintf(szBuf, "%d", i);
    TextOut(hDC, 0, y + (tm.tmHeight >> 1), szBuf, str-
    len(szBuf));
}
```

le contexte d'affichage de la fenêtre, un drapeau indiquant si l'arrière-plan de la fenêtre a été effacé et une structure de données RECT fournissant le plus petit rectangle contenant la totalité de la région de mise à jour (rectangle englobant). Si le troisième argument passé à l'une des fonctions InvalidateRect ou InvalidateRegion est TRUE, l'application spécifie que l'arrière-plan de la fenêtre doit être effacé.

Lorsque l'application appelle BeginPaint, Windows envoie le message WM_ERASEBKGND à la fenêtre, laquelle efface l'arrière-plan.

DefWindowProc répond à ce message en remplissant la zone cliente à l'aide du pinceau indiqué dans la classe de cette fenêtre (dans le champ hbrBackground de la structure WNDCLASS). La fonction BeginPaint renvoie un handle sur le contexte d'affichage de la fenêtre. L'application emploie le contexte d'affichage pour toutes les fonctions GDI qui dessine dans la zone cliente.

Lorsque BeginPaint a été appelée, l'application se charge de l'affichage et doit ensuite appeler EndPaint qui libère le contexte de périphérique.

Dans notre exemple d'application, toute la zone cliente d'une fenêtre se trouve invalidée chaque fois qu'est accomplie une opération relative aux graphes, par exemple quand une cotation est ajoutée ou quand l'un des paramètres du graphe se voit modifié, tels que le crayon ou la couleur du fond *(voir figure 13)*.

Comme nous l'avons déjà dit, GraphWndPaint est employée pour repeindre la fenêtre d'une action. Les étapes requises pour dessiner un graphe sont les suivantes :

- ❏ Définir le mapping mode, les origines et les extents pour des coordonnées cartésiennes.
- ❏ Dessiner les axes, vertical et horizontal.
- ❏ Dessiner les repères et les indications sur les axes.
- ❏ Dessiner les grilles (facultatives) verticales et horizontales.
- ❏ Tracer le graphe en fonction des prix contenus dans le tableau des cotations de l'action.

Les instructions suivantes installent le mapping mode correct :

```
/*
Définir le mapping mode.
Y positifs vers le haut, X positifs vers la
droite.
*/
SetMapMode(hDC,lpStockInfo -> graphAttrs.iMappingMode);
SetWindowExt(hDC, xGraph, yGraph);
SetViewportExt(hDC, xGraph, -yGraph);
SetViewportOrg(hDC, 0, yGraph);
```

Les extents de la fenêtre et du viewport prennent pour valeur les dimensions de la zone cliente. L'origine du viewport se trouve déplacée pour que le point (0, 0) soit situé dans le coin inférieur gauche de la fenêtre. Maintenant, les deux axes doivent être dessinés :

```
MoveTo(hDC,   yAxisStart,   xAxisStart);
LineTo(hDC,   yAxisStart,   yGraph);
MoveTo(hDC,   yAxisStart,   xAxisStart);
LineTo(hDC,   xGraph,       xAxisStart);
```

Les variables xAxisStart et yAxisStart indiquent la distance entre le début du graphe et le point (0, 0). Il faut réserver de la place sous le graphe et à gauche pour afficher les légendes horizontale et verticale. L'étape suivante consiste à dessiner les repères et les indications de chaque axe *(voir figure 11)*.

Pour l'axe des x, il faut déterminer combien de pixels sont utilisés pour chaque cotation. Cette valeur, stockée dans la variable xPixelsPerDate, représente la largeur du graphe, divisée par le nombre de cotations.

Pour chaque date affichée, l'application positionne le crayon et trace une petite ligne verticale qui servira de jalon, puis elle affiche un numéro centré sur ce repère; ce numéro ne constitue qu'une indication.

Sur l'axe des y, un repère est affiché pour chaque

FIGURE 12 : AFFICHER LES GRILLES ET LES PRIX

AFFICHER LES GRILLES

```
hPen = CreatePen(lpStockInfo->StockFile.graphinfo.iGridPen, 1,
RGB(0, 0, 0));
hOldPen = SelectObject(hDC, hPen);
if (lpStockInfo->dwFlags & STATE_HAS_VGRID)
{
    for (i = 0; i < lpStockInfo->StockFile.nTicks; i++)
    {
        int x = (i+1) * xPixelsPerDate + yAxisStart;
        MoveTo(hDC, x, xAxisStart);
        LineTo(hDC, x, yGraph);
    }
}
if (lpStockInfo->dwFlags & STATE_HAS_HGRID)
{
    for (i = rangeLow+1; i < rangeHigh; i++)
    {
        int y = (i - rangeLow) * yGraph / (rangeHigh - rangeLow);
        MoveTo(hDC, yAxisStart, y + xAxisStart);
        LineTo(hDC, xGraph, y + xAxisStart);
    }
}
SelectObject(hDC, hOldPen);
DeleteObject(hPen);
```

AFFICHER LES PRIX

```
/*
Calculer la largeur en pixels de chaque élément. Elle est
égale à la largeur du graphe divisée par le nombre de cota-
tions à afficher.
*/
  MoveTo(hDC, yAxisStart, xAxisStart + ((int)
  lpTick[0].price - rangeLow) * yGraph / (rangeHigh -
  rangeLow));
  if (lpStockInfo->StockFile.nTicks == 1)
  {
        int x = i * xPixelsPerDate;
        int y = ((int)lpTick[i].price - rangeLow) * yGraph /
                                        (rangeHigh-rangeLow);
        Ellipse(hDC, yAxisStart-2, xAxisStart-2,
        yAxisStart+2, xAxisStart+2);
  }
  else
  {
    for (i = 1; i < lpStockInfo->StockFile.nTicks; i++)
    {
        int x = i * xPixelsPerDate;
        int y = ((int)lpTick[i].price - rangeLow) * yGraph /
                                        (rangeHigh-rangeLow);
        LineTo(hDC, x + yAxisStart, y + xAxisStart);
        Ellipse(hDC, x+yAxisStart-2, y+xAxisStart-2,
                    x+yAxisStart+2, y+xAxisStart+2);
    }
}
```

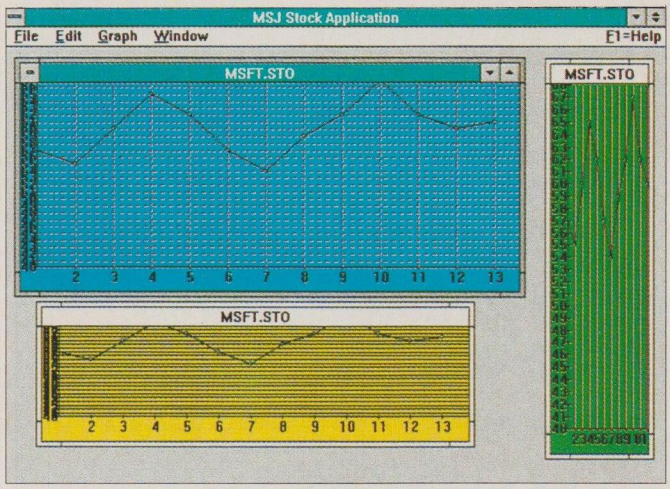


Figure 13 : les indications trop serrées sont illisibles.

valeur dans l'intervalle des valeurs de l'action. Les indications représentent ici le prix de l'action. Il faut ensuite afficher les lignes formant les grilles horizontales et verticales *(voir figure 12)*.

Remarquez l'emploi du crayon. La fonction CreatePen crée un crayon noir d'un pixel d'épaisseur. Le style est choisi par l'utilisateur dans la boîte de dialogue Graphe/Options. La fonction SelectObject associe le crayon au contexte d'affichage de la fenêtre, pour que chaque nouvelle ligne soit tracée avec le nouveau style. Le handle du crayon précédent est sauvegardé car il apparaît parfois nécessaire de s'en servir pour afficher certains éléments.

Après avoir dessiné les lignes, SelectObject est rappelée pour restaurer l'ancien crayon uni qui trace les cotations sur le graphe. Enfin, il faut appeler DeleteObject pour libérer les ressources employées par le nouveau crayon.

Arrivé à ce point du programme, la sonnette d'alarme doit résonner dans votre crâne car deux gros problèmes se posent. Tout d'abord, comment faire si le tableau de cotations contient de nombreuses dates et que la taille de la fenêtre apparaisse limitée? Les repères et les lignes verticales seront trop serrés, et les indications se chevaucheront *(voir figure 13)*. De plus, si l'intervalle est trop grand entre les valeurs minimales et maximales, ce sont les repères et les indications horizontales qui se révèleront trop serrés. C'est ici qu'il faut utiliser les facteurs d'échelle et différents intervalles de cotation.

Pour terminer, nous pouvons dessiner les cotations sur le graphe *(voir figure 12)*. Comme l'extrémité de chaque segment représente le début du suivant, il n'est pas nécessaire d'utiliser MoveTo pour déplacer le crayon d'un point à l'autre. Remarquez encore l'emploi de la fonction Ellipse pour afficher un petit cercle autour de chaque point de cotation.

Texte traduit par Henry-Luc Planchat

```
/*==================================*/
/*                                  */
/* Fichier :     GRAPH.C            */
/*                                  */
/* Description: Afficher les graphes */
/*                des actions       */
/*                                  */
/*                                  */
/* (C) Copyright 1989,1990          */
/*     Marc Adler/Magma Systems     */
/*==================================*/

#include <stdio.h>
#include <string.h>
#include <windows.h>
#include "stock.h"

void FAR PASCAL HandleSelectionState(
            LPDRAWITEMSTRUCT lpdis,
            int inflate);
void FAR PASCAL HandleFocusState(
            LPDRAWITEMSTRUCT lpdis,
            int inflate);
void FAR PASCAL DrawEntireItem(
            LPDRAWITEMSTRUCT lpdis,
            int inflate);

BOOL FAR PASCAL GraphOptionsDlgProc(hDlg,
                  message, wParam,
                  lParam)
  HWND hDlg;
  unsigned message;
  WORD wParam;
  LONG lParam;
{
  int          iPen;
  char         szBuf[80];
  static HANDLE      hStockInfo = NULL;
  static LPSTOCKINFO lpStockInfo =
(LPSTOCKINFO) NULL;
  LPGRAPHINFO lpGraphInfo;
  HWND         hWnd;
  BOOL         bTranslated;


  switch (message)
  {
    case WM_INITDIALOG :
      if ((hStockInfo =
            (HANDLE) lParam) == NULL)
      {
        EndDialog(hDlg, IDCANCEL);
        return TRUE;
      }

      if ((lpStockInfo =
            (LPSTOCKINFO)
            GlobalLock(hStockInfo))
            == NULL)
      {
        EndDialog(hDlg, IDCANCEL);
        return TRUE;
      }

      lpGraphInfo = &lpStockInfo->
            StockFile.graphinfo;



      /*
      Si une zone mémoire est allouée au
      tableau des cotations, cela
      signifie que nous travaillons
      avec un tableau existant. Sinon
      nous créons une nouvelle action.
      */


      if (lpStockInfo->hTicks != NULL)
      {
        SetDlgItemText(hDlg, ID_SYMBOL,
            lpStockInfo->
            StockFile.szStock);
        SetDlgItemInt(hDlg,ID_MINPRICE,
           (int) lpGraphInfo->
            dwMinPrice, FALSE);
        SetDlgItemInt(hDlg,ID_MAXPRICE,
           (int) lpGraphInfo->
            dwMaxPrice, FALSE);
        SetDlgItemInt(hDlg, ID_FACTOR,
           (int)lpGraphInfo->
            dwScaleFactor, FALSE);
        SetDlgItemInt(hDlg, ID_TICKINT,
          (int)lpGraphInfo->
            dwTickInterval, FALSE);
        SetDlgItemInt(hDlg,
            ID_DENOMINATOR,
          (int) lpGraphInfo->
            iDenominator, FALSE);

        if (lpStockInfo->dwFlags &
            STATE_HAS_VGRID)
            CheckDlgButton(hDlg,
            ID_VERTGRID, TRUE);
        if (lpStockInfo->dwFlags &
            STATE_HAS_HGRID)
            CheckDlgButton(hDlg,
            ID_HORZGRID, TRUE);

        EnableWindow(GetDlgItem(hDlg,
            ID_SYMBOL), FALSE);
      }
      else
      {
        lpGraphInfo->iGridPen = 0;
      }



      /*
      Remplir la boîte liste avec les
      identificateurs de crayon
      */
      for (iPen = 0;
           iPen < PS_NULL;
           iPen++)
        SendDlgItemMessage(hDlg,
         ID_GRIDSTYLE, CB_ADDSTRING,0,
            (LONG) iPen);
      SendDlgItemMessage(hDlg,
      ID_GRIDSTYLE,CB_SETCURSEL,
      lpGraphInfo->iGridPen,0L);
      return TRUE;

    case WM_MEASUREITEM :
    {
      LPMEASUREITEMSTRUCT lpMI =
            (LPMEASUREITEMSTRUCT)
            (LPSTR)lParam;
      lpMI->itemWidth  = 42;
      lpMI->itemHeight = 20;
      break;
    }
```

```
case WM_DRAWITEM :
{
  LPDRAWITEMSTRUCT lpdis =
        (LPDRAWITEMSTRUCT) (LPSTR)
        lParam;

  if (lpdis->itemID == -1)
  {
    HandleFocusState(lpdis, -5);
    break;
  }

  switch (lpdis->itemAction)
  {
    case ODA_DRAWENTIRE:
      DrawEntireItem(lpdis, -4);
      break;

    case ODA_SELECT:
     HandleSelectionState(lpdis,0);
     break;

    case ODA_FOCUS:
      HandleFocusState(lpdis, -2);
      break;
  }

  break;
}

case WM_COMMAND :
  switch (wParam)
  {
    case IDOK :
     lpGraphInfo = &lpStockInfo->
        StockFile.graphinfo;

      if (lpStockInfo->hTicks ==
  NULL)
      {
        GetDlgItemText(hDlg,
          ID_SYMBOL,
          szBuf, sizeof(szBuf));
        if (*szBuf == '\0')
        {
          MessageBox(hWndMain,
            "Vous devez donner le
            nom de l'action.",
            "Error", MB_OK);
          return TRUE;
        }
        strcat(szBuf, ".STO");
        lstrcpy(
          lpStockInfo->szFileName,
          szBuf);

        lpStockInfo->
          StockFile.dwMagic
          = MAGIC_COOKIE;
        if ((lpStockInfo->hTicks =
         GlobalAlloc(GMEM_MOVEABLE,
         (DWORD) sizeof(TICK)*64))
         == NULL)
        {
          MessageBox(hWndMain,
            "Impossible d'allouer
            le tableau de
            cotation",
            "Error",MB_OK);
          goto bye;
        }
        lpStockInfo->
          nTicksAllocated = 64;
        lpStockInfo->
          StockFile.nTicks = 0;



        /*
          Créer une fenêtre
        */
        hWnd = lpStockInfo->hWnd =
          GraphCreateWindow(
          (LPSTR) szBuf);
        SetWindowWord(
          lpStockInfo->hWnd, 0,
          hStockInfo);
        lpStockInfo->dwFlags |=
          STATE_DIRTY;
      }

      GetDlgItemText(hDlg,
        ID_SYMBOL, lpStockInfo->
        StockFile.szStock,
        sizeof(lpStockInfo->
        StockFile.szStock));
      lpGraphInfo->dwMinPrice =
        GetDlgItemLong(hDlg,
        ID_MINPRICE,
        (BOOL FAR *) &bTranslated,
        FALSE);
      lpGraphInfo->dwMaxPrice =
        GetDlgItemLong(hDlg,
        ID_MAXPRICE,
        (BOOL FAR *) &bTranslated,
        FALSE);
      lpGraphInfo->dwScaleFactor =
        GetDlgItemLong(hDlg,
        ID_FACTOR,
        (BOOL FAR *) &bTranslated,
        FALSE);
      lpGraphInfo->dwTickInterval =
        GetDlgItemLong(hDlg,
        ID_TICKINT,
        (BOOL FAR *) &bTranslated,
        FALSE);
      lpGraphInfo->iDenominator =
        GetDlgItemInt(hDlg,
        ID_DENOMINATOR,
        (BOOL FAR *) &bTranslated,
        FALSE);
      lpGraphInfo->iGridPen =
        (WORD) SendDlgItemMessage(
        hDlg,ID_GRIDSTYLE,
        CB_GETCURSEL,0,0L);

      lpStockInfo->dwFlags &=
        ~(STATE_HAS_VGRID|
        STATE_HAS_HGRID);
      if (IsDlgButtonChecked(hDlg,
        ID_VERTGRID))
        lpStockInfo->dwFlags |=
        STATE_HAS_VGRID;
      if (IsDlgButtonChecked(hDlg,
        ID_HORZGRID))
        lpStockInfo->dwFlags |=
STATE_HAS_HGRID;
```

```
bye:
          InvalidateRect(lpStockInfo->
            hWnd, (LPRECT) NULL,FALSE);
          GlobalUnlock(hStockInfo);
          EndDialog(hDlg, IDOK);
          break;


        case IDCANCEL:
          GlobalUnlock(hStockInfo);
          EndDialog(hDlg, IDCANCEL);
          break;
      }
      return TRUE;

  } /* end switch (message) */

  return FALSE;
}


/************************************
Fonction    :
HandleSelectionState(LPDRAWITEMSTRUCT,
  int)
Description : Gère une modification
  dans le statut de sélection d'un
  élément. Si un élément est
  sélectionné, un cadre
  rectangulaire noir entoure l'élément;
  si l'élément est désélectionné, le
  cadre est supprimé.
Commentaire : Le cadre de sélection noir
  est un peu plus grand que le cadre grisé
  du focus pour éviter les confusions.
************************************/
void FAR PASCAL HandleSelectionState(
  lpdis, inflate)
  LPDRAWITEMSTRUCT  lpdis;
  int               inflate;
{
  HBRUSH hbr =
    (lpdis->itemState & ODS_SELECTED) ?

      GetStockObject(BLACK_BRUSH)     :
      CreateSolidBrush(
      GetSysColor(COLOR_WINDOW));
FrameRect(lpdis->hDC, (LPRECT)
    &lpdis->rcItem, hbr);
  DeleteObject (hbr);
}
/************************************
Fonction    :
HandleFocusState(LPDRAWITEMSTRUCT, int)
Description :
  Gère une modification du statut de
  focus d'un élément. Si un élément
  obtient le focus, un cadre
  rectangulaire grisé est dessiné
  autour de l'élément; si un élément
  perd le focus, le cadre est effacé.
Commentaire :
  Le cadre de focus grisé est
  légèrement plus petit que le cadre
  de sélection noir pour éviter les
  confusions.
************************************/
void FAR PASCAL HandleFocusState(lpdis,
inflate)
  LPDRAWITEMSTRUCT  lpdis;
  int       inflate;
{
  RECT  rc;
  HBRUSH  hbr;

  /* Redéfinir la taille du rectangle
     pour placer le cadre de focus
     entre le cadre de sélection et
     l'élément.
   */
  CopyRect ((LPRECT)&rc,
    (LPRECT)&lpdis->rcItem);
  InflateRect
    ((LPRECT)&rc, inflate, inflate);

  if (lpdis->itemState & ODS_FOCUS)
    /* le focus est atribué -- dessiner
       un cadre gris */
    hbr = GetStockObject(GRAY_BRUSH);
  else
    /* le focus est perdu -- supprimer
       le cadre */
    hbr = CreateSolidBrush(
          GetSysColor(COLOR_WINDOW));

  FrameRect(lpdis->hDC, (LPRECT)&rc,
      hbr);
  DeleteObject (hbr);
}

/************************************
Fonction    :
DrawEntireItem(LPDRAWITEMSTRUCT, int)
Description :
  Dessine un élément et l'entoure d'un
  cadre de sélection et/ou d'un cadre
  de focus en cas de besoin.
************************************/
void FAR PASCAL DrawEntireItem(lpdis,
  inflate)
  LPDRAWITEMSTRUCT  lpdis;
  int       inflate;
{
  RECT   rc;
  HANDLE hOldPen;
  HPEN   hPen;

  if (lpdis->itemData < 0)
    return;

  /* Redéfinir le rectangle afin de
     laisser de la place pour les
     cadres */
  CopyRect ((LPRECT)&rc,
    (LPRECT)&lpdis->rcItem);
  InflateRect
    ((LPRECT)&rc, inflate, inflate);

  if (lpdis->itemState & ODS_FOCUS)
    hPen = CreatePen((int)lpdis->
           itemData, 1,
           RGB(0xFF, 0xFF, 0xFF));
  else
    hPen = CreatePen((int) lpdis->
           itemData, 1, RGB(0, 0, 0));
  hOldPen =
    SelectObject(lpdis->hDC, hPen);

  MoveTo(lpdis->hDC, rc.left,
    rc.top + (rc.bottom - rc.top) / 2);
  LineTo(lpdis->hDC, rc.right,
    rc.top + (rc.bottom - rc.top) / 2);
```

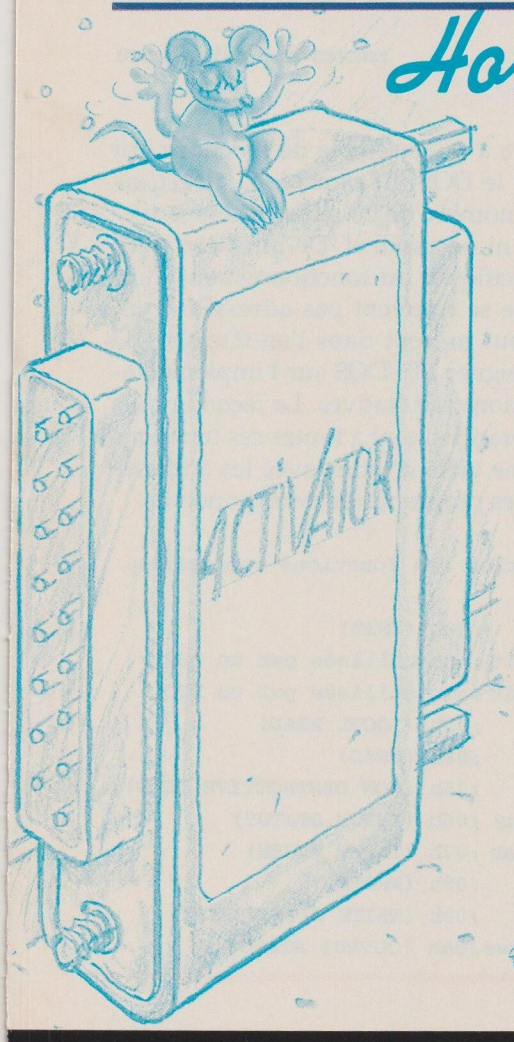
```
  SelectObject(lpdis->hDC, hOldPen);
  DeleteObject(hPen);

  /* Afficher ou effacer les cadres
     appropriés */
  HandleSelectionState(lpdis,
     inflate + 4);
  HandleFocusState(lpdis, inflate + 2);
}


int PASCAL GraphWndPaint(HWND hWnd,
       HDC hDC,
            LPSTOCKINFO lpStockInfo)
{
  RECT          r;
  int           i, x, y;
  int           xGraph, yGraph;
  LPTICK        lpTick;
  int           rangeHigh,
                rangeLow,
                xPixelsPerDate;
  int           yAxisStart, xAxisStart;
  HANDLE        hOldPen;
  HPEN          hPen;
  HBRUSH        hBrush;
  TEXTMETRIC    tm;
  char          szBuf[80];

  if (lpStockInfo->StockFile.nTicks ==
    0)
    return TRUE;

/*
  Récupérer les dimensions de la zone
  cliente de la fenêtre de l'action
  courante et déterminer la largeur
  et la hauteur du graphe.
*/
GetClientRect(hWnd, (LPRECT) &r);
yGraph = r.bottom - r.top;
xGraph = r.right - r.left;

/*
  Effacer l'arrière-plan avec la
  couleur voulue
*/
hBrush = CreateSolidBrush(lpStockInfo->
    graphAttrs.clrBackground);
FillRect(hDC, (LPRECT) &r, hBrush);
DeleteObject(hBrush);

/*
  Placer les intervalles dans des
  variables locales.
*/
rangeHigh = (int) lpStockInfo->
  StockFile.graphinfo.dwMaxPrice;
rangeLow  = (int) lpStockInfo->
  StockFile.graphinfo.dwMinPrice;

/*
  Déterminer la taille de la police
  courante.
*/
GetTextMetrics(hDC,
  (LPTEXTMETRIC) &tm);

if ((lpTick = (LPTICK)
  GlobalLock(lpStockInfo->hTicks)) ==
  NULL)
  return FALSE;
```

```
/*
  Définir le mapping mode.
  L'axe des Y pointe vers le haut.
  L'axe des X pointe vers la droite.
*/
SetMapMode(hDC, lpStockInfo->
  graphAttrs.iMappingMode);
SetWindowExt(hDC,  xGraph, yGraph);
SetViewportExt(hDC, xGraph, -yGraph);
SetViewportOrg(hDC, 0, yGraph);

sprintf(szBuf, "%d", rangeHigh);
yAxisStart = strlen(szBuf) *
  (tm.tmAveCharWidth + 1);
xAxisStart =
  tm.tmHeight + (tm.tmHeight/2);
xPixelsPerDate = xGraph /
 lpStockInfo->StockFile.nTicks;

/*
  Donner l'attribut TRANSPARENT au
  mode d'arrière-plan pour remplir
  les espaces libres avec la couleur
  d'arrière-plan.
*/
SetBkMode(hDC, TRANSPARENT);

/*
  Afficher les axes. L'axe des Y en
  premier.
*/
MoveTo(hDC, yAxisStart, xAxisStart);
LineTo(hDC, yAxisStart, yGraph);
MoveTo(hDC, yAxisStart, xAxisStart);
LineTo(hDC, xGraph,     xAxisStart);

/*
  Afficher les repères de l'axe des x
*/
for (i = 1;
  i < lpStockInfo->StockFile.nTicks;
  i++)
{
  int x = i * xPixelsPerDate +
    yAxisStart;
  int iTextLen;
  MoveTo(hDC, x, xAxisStart-2);
  LineTo(hDC, x, xAxisStart+2);
  sprintf(szBuf, "%d", i+1);
  iTextLen = strlen(szBuf) *
    tm.tmAveCharWidth;
  /*
    Centrer le texte sur le repère
  */
  TextOut(hDC, x-(iTextLen>>1),
    xAxisStart-2,
    szBuf,strlen(szBuf));
}

/*
  Afficher les repères de l'axe des y
*/
for (i = rangeLow;
  i < rangeHigh;  i++)
{
  int y = (i-rangeLow) *
    yGraph/(rangeHigh-rangeLow) +
    xAxisStart;
  MoveTo(hDC, yAxisStart-4, y);
  LineTo(hDC, yAxisStart+4, y);
  sprintf(szBuf, "%d", i);
```

```
    TextOut(hDC, 0,
      y + (tm.tmHeight >> 1), szBuf,
      strlen(szBuf));
}


/*
  Afficher maintenant les grilles
  horizontales et verticales.
*/
hPen = CreatePen(lpStockInfo->
  StockFile.graphinfo.iGridPen, 1,
  lpStockInfo->graphAttrs.clrPen);
hOldPen = SelectObject(hDC, hPen);

if (lpStockInfo->dwFlags &
   STATE_HAS_VGRID)
{
  for (i = 0;  i < lpStockInfo->
    StockFile.nTicks;  i++)
  {
    int x = (i+1) * xPixelsPerDate +
      yAxisStart;
    MoveTo(hDC, x, xAxisStart);
    LineTo(hDC, x, yGraph);
  }
}


if (lpStockInfo->dwFlags &
  STATE_HAS_HGRID)
{
  for (i = rangeLow+1;
    i < rangeHigh;  i++)
  {
    int y = (i - rangeLow) * yGraph /
      (rangeHigh - rangeLow);
    MoveTo(hDC, yAxisStart,
      y + xAxisStart);
    LineTo(hDC, xGraph,
      y + xAxisStart);
  }
}

SelectObject(hDC, hOldPen);
DeleteObject(hPen);


/*
  Calculer la largeur en pixels de
  chaque élément. Elle est égale
  à la largeur du graphe divisée par
  le nombre de cotations à afficher.
*/
MoveTo(hDC, yAxisStart, xAxisStart +
  ((int) lpTick[0].price - rangeLow)
  * yGraph / (rangeHigh - rangeLow));
if (lpStockInfo->StockFile.nTicks==1)
{
  Ellipse(hDC,
    yAxisStart-2, xAxisStart-2,
    yAxisStart+2, xAxisStart+2);
}
else
{
  for (i = 1;  i < lpStockInfo->
    StockFile.nTicks;  i++)
  {
    int x = i * xPixelsPerDate;
    int y = ((int)lpTick[i].price -
      rangeLow) * yGraph /
      (rangeHigh-rangeLow);
    LineTo(hDC, x + yAxisStart,
      y + xAxisStart);
    Ellipse(hDC,
     x+yAxisStart-2, y+xAxisStart-2,
     x+yAxisStart+2, y+xAxisStart+2);
  }
 }


bye:
  GlobalUnlock(lpStockInfo->hTicks);
  return TRUE;
}


#include "rainbow.h"

BOOL FAR PASCAL GraphColorsDlgProc(hDlg,
msg, wParam, lParam)
  HWND hDlg;
  WORD msg;
  WORD wParam;
  LONG lParam;
{
  LPSTOCKINFO lpStockInfo;
  char        szMsg[80];

  switch (msg)
  {
    case WM_INITDIALOG:
      /*
        S'assurer qu'il existe un
        enregistrement d'information
        auquel on puisse ajouter la
        cotation.
      */
      if (!hCurrStockInfo)
      {
        MessageBox(hDlg, "Il n'y a pas
          encore de graphe de
          l'action.",
          "Error", MB_OK);
        EndDialog(hDlg, FALSE);
        return TRUE;
      }

      lpStockInfo = (LPSTOCKINFO)
        GlobalLock(hCurrStockInfo);
      if (lpStockInfo == NULL)
      {
        MessageBox(hDlg,
          "GlobalLock a renvoyé NULL",
          "Error", MB_OK);
        EndDialog(hDlg, FALSE);
        return TRUE;
      }

      SendDlgItemMessage(hDlg,
        ID_PENCOLOR,  RM_SETSEL,
        lpStockInfo->graphAttrs.idxPen,
        0L);
      SendDlgItemMessage(hDlg,
        ID_BACKCOLOR, RM_SETSEL,
        lpStockInfo->
        graphAttrs.idxBackground, 0L);

      GlobalUnlock(hCurrStockInfo);
      return TRUE;
```

```
case WM_COMMAND:
  switch (wParam)
  {
    /*
      L'utilisateur a choisi le
      bouton OK...
    */
    case IDOK:
      /*
        Obtenir un pointeur sur
        l'enregistrement
        d'information
      */
      lpStockInfo = (LPSTOCKINFO)
        GlobalLock(hCurrStockInfo);
      if (lpStockInfo == NULL)
      {
        MessageBox(hDlg,
          "GlobalLock a renvoyé
          NULL", "Error", MB_OK);
        EndDialog(hDlg, FALSE);
        break;
      }

      lpStockInfo->
        graphAttrs.idxPen = (WORD)
        SendDlgItemMessage(
        hDlg, ID_PENCOLOR,
        RM_GETSEL, 0,
        (LONG) (LPSTR)
        &lpStockInfo->
          graphAttrs.clrPen);
        lpStockInfo->
          graphAttrs.idxBackground =
          (WORD)
          SendDlgItemMessage(hDlg,
          ID_BACKCOLOR, RM_GETSEL, 0,
          (LONG) (LPSTR)
          &lpStockInfo->
          graphAttrs.clrBackground);

        GlobalUnlock(hCurrStockInfo);
        InvalidateRect(
          lpStockInfo->hWnd,
          (LPRECT) NULL, FALSE);
        EndDialog(hDlg, TRUE);
        break;


      /*
        L'utilisateur a choisi le
        bouton ANNULER...
      */
      case IDCANCEL :
        EndDialog(hDlg, FALSE);
        break;
    }
    return TRUE;

  default:
    return FALSE;
  }
}
```


Hors de portée des convoitises !!!



SÉLECTIONNÉE PAR **MICRO SIGMA** LA GAMME ACTIVATOR

REPOND A VOS BESOINS SPECIFIQUES DE PROTECTION INFORMATIQUE

- Contrôle d'accès à des programmes et/ou modules par mots de passe
- Contrôle du nombre d'utilisateurs pouvant simultanément accéder à un même programme
- Contrôles programmables à partir d'informations stockées dans une mémoire non volatile
- Environnements : DOS, UNIX, MAC, SUN...
- A partir de 250 Frs. H.T.

Consultez-nous au 46 22 99 88 Fax : 47 63 84 42

MICRO SIGMA Outils de Développement
42, rue Legendre - 75017 PARIS

Service lecteurs n° 9 (dernière page)

ACTIVATOR est une marque déposée de SOFWARE SECURITY - USA

ATOR ACTIVATOR/N ACTIVATOR/S ACTIVATOR/M MACTIVATOR ACTIVATOR ACT

SYSTÈME

DRIVER SOUS DOS

Écriture d'un pilote logiciel pour port série

JEAN-FRANÇOIS DUVIVIER

Le système MS-DOS, à partir de sa version 2.0, offre la possibilité de rajouter de façon simple de nouveaux gestionnaires de périphériques. Ces programmes, appelés «device drivers» (notés DD dans la suite de cet article), s'intègrent étroitement au système MS-DOS dont ils deviennent partie prenante.

En effet, le fichier IO.SYS du DOS comprend déjà des DD résidents (CON, PRN, AUX, NUL, CLOCK...), et les DD chargés à partir du fichier Config.SYS pourront soit les remplacer, soit s'y ajouter. Du fait de l'imbrication étroite avec le DOS, le format et l'appel de ces DD sont spécifiques et s'éloignent des exécutables classiques .COM ou .EXE.

Il existe deux formes de DD. Ceux de type «bloc» gèrent et transfèrent leurs données par bloc (sur disque, disquette, bande...). Ils sont identifiés par des lettres. A: permet d'accéder au driver résident qui gère le premier lecteur de disquettes. Les DD de type «caractère» (notés DDC par la suite) transfèrent leurs données caractère par caractère (par exemple, sur écran ou à partir du clavier). Les DDC portent un nom qui les identifie et qui autorise l'accès aux programmes d'application avec la même syntaxe que pour un fichier. Un DD comporte un en-tête spécifique qui permet au DOS de connaître son nom, ses spécificités et ses points d'entrées. Notre DD est de type caractère, porte le nom d'ASYN et supporte deux types de fonctions facultatives : les lectures-écritures IOCTL et les «Open/Close». Les deux premiers mots se trouvent initialisés à 0FFFFh mais seront modifiés lors de l'initialisation par le MS-DOS avec l'adresse du DD précédent.

```
.MODEL SMALL
.CODE
                org     0
;—— En-tête obligatoire du device driver
dw  -1,-1               ;dernier DD
dw  1100100000000000b ;attributs
dw  strat               ;routine de stratégie
dw  intdd               ;routine d'interruption
db  'ASYN    '          ;Nom logique du périphérique
```

La routine de stratégie est la première appelée, par l'intermédiaire d'un Call Far, avec l'adresse complète d'un bloc de requêtes dans ES:BX. Cette routine stocke l'adresse pour une utilisation ultérieure et rend la main immédiatement. Elle semble donc très simple et très courte.

```
;—— Point d'entrée de la routine de stratégie
ddbloc  dd  0                   ;Bloc de requête
strat:  mov cs:[ddbloc],bx      ;sauvegarde de l'offset
        mov cs:[ddbloc+2],es    ;sauvegarde du segment
stratp  proc far
        ret                     ;retour far obligatoire
stratp  endp
```

Le DD doit répondre à un ensemble de fonctions qui lui sont adressées par le DOS *(Cf encadré : Les fonctions du DD)*. Elles sont au nombre de 20 mais elles ne s'avèrent pas toutes utiles ni nécessaires. De plus, certaines d'entre elles sont spécifiques au fonctionnement d'un DD de type bloc et ne se trouvent pas adressées à un DDC. Le mot d'attribut présent dans l'en-tête du DD sert également à renseigner MS-DOS sur l'implémentation de certaines fonctions facultatives. La façon la plus simple d'effectuer le branchement à toutes ces fonctions consiste à préparer une table d'index avec les adresses des points d'entrées des programmes correspondants.

```
;—— Table d'indexation des fonctions supportées
cmdtbl:
        dw init         ;00h (INIT)
        dw cmderr_exit  ;non utilisée par un DDC
        dw cmderr_exit  ;non utilisée par un DDC
        dw ioctl_in     ;03h (IOCTL READ)
        dw read         ;04h (READ)
        dw nd_read      ;05h (NON-DESTRUCTIVE READ)
        dw input_status ;06h (INPUT STATUS)
        dw input_flush  ;07h (INPUT FLUSH)
        dw write        ;08h (WRITE)
        dw write        ;09h (WRITE WITH VERIFY)
        dw output_status;0Ah (OUTPUT STATUS)
```

Certains aspects du MS-DOS restent méconnus et peu utilisés par les développeurs, principalement en raison de leur complexité supposée. C'est le cas des pilotes logiciels ou «device drivers» qui sont entourés d'un halo mystérieux et pour lesquels peu de programmeurs ont entrepris le développement. Il s'agit pourtant de notions simples et faciles à mettre en œuvre. C'est ce que nous avons voulu montrer dans un exemple d'inspiration pédagogique qui aborde également la gestion du port série sous interruption.

```
        dw output_flush ;0Bh (OUTPUT FLUSH)
        dw ioctl_out    ;0Ch (IOCTL WRITE)
        dw open_dev     ;0Dh (OPEN DEVICE)
        dw close_dev    ;0Eh (CLOSE DEVICE)
        dw cmderr_exit  ;non utilisée
        dw write        ;10h (OUTPUT UNTIL BUSY)
```

Le bloc de requête stocké par la routine de stratégie constitue le seul lien du DD avec le DOS. Il contient l'ensemble des informations nécessaires au traitement d'une fonction ainsi que les résultats de l'opération. On apprécie de définir quelques équivalences pour les informations qui seront le plus souvent accédées :

```
;—— Equivalence des positions de certaines
;—— informations par rapport au début du bloc de
;—— requête
fonction        = 2     ;numéro de fonction
status          = 3     ;status
transfert       = 14    ;adresse du buffer
nbcar           = 18    ;nombre de car. à traiter
```

Lorsque le DOS appelle un DD, le registre de segment SS n'est pas commuté sur une pile locale. Le DD aura donc accès à la pile du DOS dont la taille se trouve limitée. Pour cette raison, mais aussi pour des motifs de sécurité, il est souhaitable de ne pas l'utiliser. Il est donc nécessaire de prévoir une pile locale en fonction des besoins du DD et d'effectuer une commutation de pile dès le début de l'exécution de ce programme.

```
;—— Données nécessaires pour le DD (pile locale)
savss   dw 0            ;Sauvegarde segment de pile
savsp   dw 0            ;Sauvegarde offset de pile
        dw 100 dup (0) ;Pile locale
piledd  label word      ;définition fin de pile
```

La routine d'interruption est donc appelée par le MS-DOS pour traiter le bloc de requête. Le nom donné à cette routine se révèle totalement inapproprié puisqu'elle n'est pas appelée par une interruption (hard IRQ ou soft INT) mais par un Call Far. La notion de deux points d'entrées pour un même DD n'est actuellement d'aucune utilité à MS-DOS qui appelle la routine d'interruption directement après celle de stratégie. Mais elle avait été pensée au départ (en 1983!) pour les versions «futures et multitâches» du DOS, afin d'autoriser un asynchronisme entre la requête et son traitement. La première mission du DD sera de commuter le segment de pile sur une pile locale, puis d'effectuer une sauvegarde de tous les registres utilisés, y compris les flags.

```
;—— Point d'entrée de la routine  ;
d'interruption
intdd: pushf                    ;sauvegarde des flags
       cli                      ;dévalidation
mov cs:word ptr savss,ss        ;des interruptions
mov cs:word ptr savsp,sp        ;pendant la sauvegarde
mov sp,cs                       ;des registres de pile
mov ss,sp                       ;et la commutation
mov sp,offset piledd            ;sur la pile locale
sti
push ax                         ;sauvegarde de tous
push bx                         ;les registres
push cx
push dx
push si
push di
push bp
push ds
push es
```

Le programme peut ensuite rechercher dans le bloc de requête, à l'adresse stockée par la routine de stratégie, le numéro de la fonction à exécuter, lequel servira d'index dans la table des fonctions. Celle-ci se limite aux seize premières fonctions, et tout numéro supérieur à 16 génère une erreur de type «Commande inconnue».

Le nombre de caractères et l'adresse de transfert, deux paramètres souvent utilisés par les diverses fonctions, sont ensuite chargés dans deux registres. Celui de segment de donnée se recharge avec le segment de

code courant, afin d'accéder plus facilement aux variables locales au DD. Le jump final permet le branchement direct à la fonction demandée.

```
lds bx,cs:[ddbloc]     ;adresse bloc de requête
mov al,[bx].fonction   ;numéro de fonction
cbw                    ;extension octet -> mot
mov si,offset cmdtbl   ;table d'indexation
add si,ax              ;pour le traitement des
add si,ax              ;différentes fonctions
cmp al,16              ;avec test de débordement
ja cmderr_exit         ;commande inconnue
mov cx,[bx].nbcar      ;nombre de caractères
les di,[bx].transfert ;adr transfert dans es:di
push cs
pop ds                 ;ds=cs
cld
jmp word ptr [si]      ;exécution de la fonction
```

La sortie du DD doit toujours s'effectuer avec le bit 8 positionné dans le mot de status (il indique que le bloc de requête a été traité), mais aussi parfois avec le bit 9 («busy») ou 15 («erreur»).

Le programme de sortie aura donc à mettre à jour le status suivant les différentes possibilités puis devra restaurer les registres et recommuter la pile sur sa valeur initiale. Le retour au MS-DOS se fait par un renvoi de type FAR.

```
;—— Sortie du device driver avec gestion des
;—— différentes erreurs
busy_exit:      mov ax,0000001100000000b
                        ;sortie avec bit busy
                jmp short exit
                        ;positionné
err_exit:       mov ah,10000001b
                        ;sortie avec bit erreur
                jmp short exit
                        ;positionné
cmderr_exit:    mov ax,1000000100000011b
                        ;sortie avec bit erreur
                jmp short exit
                        ;et "commande inconnue"
ok_exit:        mov ax,0000000100000000b
                        ;sortie sans erreur
exit:           lds bx,cs:[ddbloc]
                        ;adresse bloc de requête
                mov [bx].status,ax
                        ;mise à jour du status
        pop es          ;restauration
        pop ds          ;de tous les
        pop bp          ;registres
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax
        cli             ;dévalidation des int quand
        mov ss,cs:savss ;on restaure la pile
        mov sp,cs:savsp ;d'origine
        popf            ;restauration des flags
exitp proc      far
                ret     ;retour far obligatoire
exitp endp
```

La possibilité d'utilisation d'un port série à une adresse quelconque, associé à un niveau d'interruption variable (3, 4 ou autre), impose de recréer à chaque nouvelle configuration les deux masques nécessaires au contrôleur d'interruption (PIC 8259) pour l'autorisation d'une IRQ (bit i positionné à 1 pour l'IRQi) et l'acquittement de l'IRQ en fin de traitement (valeur 60h + numéro de l'IRQ). De plus, le vecteur d'interruption correspondant à l'IRQ (soit l'int 8 + numéro de l'IRQ) sera sauvegardé afin de pouvoir restaurer sa valeur initiale.

```
;—— Données locales nécessaires au traitement
;—— du port série
basio   dw 0            ;adresse du port COM
nivit   dw 0            ;08h + IRQ
msit    db 0            ;masque autorisation IRQ
ackit   db 0            ;masque acquittement IRQ
old_veccom dd 0         ;ancien vecteur IRQ
```

Si la gestion sous interruption apporte d'énormes avantages dans l'utilisation d'un port série (en particulier l'absence de pertes de caractères), elle a pour contrepartie de réclamer des buffers internes pour stocker ces caractères en attente de la prise en compte par l'application.

Ces buffers, l'un en entrée pour les caractères reçus, l'autre en sortie pour ceux en attente d'envoi, doivent être suffisamment dimensionnés pour éviter toute perte de données. Leur gestion requiert également une grande simplicité et la meilleure performance.

Le buffer circulaire, avec pointeurs de lecture et d'écriture semble tout indiqué. L'ajout du nombre de caractères maintenus dans le buffer est intéressant, car il permet de simplifier tous les tests de présence-absence de ceux-ci, ainsi que les transferts de caractères multiples.

```
;—— Définition des buffers utilisés en
;    entrée et en sortie
bufin_lg  equ 1000          ;taille buffer
                            ;d'entrée
bufin_nb  dw 0              ;nb car. dans
                            ;le buffer
```

```
bufin_rd  dw offset bufin  ;pointeur de
                           ;lecture
bufin_wr  dw offset bufin  ;pointeur
                           ;d'écriture
bufin     db bufin_lg dup (0)
bufin_end dw 0             ;fin du buffer
bufout_lg equ 200          ;taille buffer
                           ;de sortie
bufout_nb dw 0             ;nb car. dans
                           ;le buffer
bufout_rd dw offset bufout ;pointeur de
                           ;lecture
bufout_wr dw offset bufout ;pointeur
                           ;d'écriture
bufout    db bufout_lg dup (0)
bufout_end dw 0             ;fin du buffer
```

La définition du port série utilisé, de l'IRQ choisie et des paramètres de communications doivent être bien évidemment configurables. Afin d'avoir le maximum de souplesse, ces données impliquent de pouvoir être modifiées directement par un programme d'application à partir des commandes IOCTL (int 21h / 4402h et 4403h). Les données fournies lors de ces appels sont transférées directement par le MS-DOS au DD. Leur format dépend du DD lui-même. Dans notre cas, 7 octets suffisent à définir les paramètres variables.

```
;—— Données relatives à la voie série
;      et faisant partie des données de
;      contrôle du device driver
ioctl_parm     label          byte
comadr         dw 3F8h        ;adresse par
                              ;défaut du port
comirq         dw 4           ;IRQ utilisée par
                              ;défaut
dvl            dw 0Ch         ;9600 bauds par défaut
fcar           db 00000011b   ;8 bits,1 stop, pas
                              ;de parité
ioctl_parm     label          byte
```

Les deux fonctions 03h et 0Ch permettent de lire et d'écrire la zone précédemment définie. Elles utilisent

Une clé Microphar ne protège pas seulement vos logiciels. Elle vous protège de bien d'autres choses.

Expérience : Depuis 1983, 650 000 clés Microphar protègent les logiciels de 3 500 SSII et grandes entreprises - **Sécurité :** chaque client bénéficie d'une fabrication personnalisée, réalisée uniquement à sa demande - **Evolution :** les routines d'interrogation des clés sont maintenues dans plus de 130 langages sous DOS, UNIX, OS/2 et WINDOWS - **Pérennité :** 22 personnes assurent l'assistance-client et une innovation constante : clé électronique et clé à mémoire pour PC et PS, clé pour Macintosh, clé à microprocesseur pour micros, minis, stations de travail et tout ordinateur disposant d'un port série RS 232. **Nouveau :** notre dernière clé à microprocesseur dispose d'une mémoire programmable qui autorise de surcroit le contrôle d'accès. Aujourd'hui, Microphar exporte 55 % de sa production vers 11 pays d'Europe et aux USA.

MICROPHAR

n°1 européen de la protection de logiciels

122, avenue Charles de Gaulle - 92200 Neuilly/Seine - Tél.: (1) 47 38 21 21 - Fax : (1) 46 24 76 91 - Belgique : 091 21 11 17 - Suisse : 024 21 53 86

une partie de code commun qui autorise la vérification de la validité du nombre de caractères à traiter, ainsi que le transfert lui-même, effectué avec les opérations de traitement de chaîne MOVSB. Seule la direction du transfert change entre ces deux fonctions, et donc la valeur des registres ES:DI et DS:SI.

```
;—— Point d'entrée de la fonction 0Ch
;—— (IOCTL Write)
ioctl_out:
     push es           ;passage ES:DI dans DS:SI
     pop  ds
     mov  si,di
     push cs           ;chargement ES:DI par
     pop  es           ;l'adresse des paramètres
     lea di,ioctl_parm ;configurables
     jmp short ioctl

;—— Point d'entrée de la fonction 03h
;—— (IOCTL Read)
ioctl_in:
     push cs          ;chargement DS:SI par
     pop  ds          ;l'adresse des paramètres
     lea  si,ioctl_parm ;configurables

;—— Transfert des paramètres ioctl:
     cmp cx,ioctl_parme - ioctl_parm
     je  iout1      ;vérif du nbre d'octets
     mov al,0Ch     ;erreur General failure
     jmp err_exit   ;sortie en erreur
iout1: rep movsb    ;transfert des
                    ;paramètres
     jmp ok_exit    ;sortie normale
```

La lecture des caractères reçus par le driver s'effectue, à partir d'une application, par le biais de l'int 21h/3Fh qui permet de lire indistinctement un fichier ou un périphérique.

Le transfert des caractères dans le buffer fourni par le programme appelant, se fait en tenant compte du nombre de caractères effectivement disponibles, mais aussi du nombre de caractères demandé par l'application. La quantité réelle de caractères transférés doit être mise à jour dans le bloc de requête.

```
;—— Point d'entrée de la fonction 04h (READ)
read:  push cx     ;nombre car. max à
                   ;traiter
read1: call rcar   ;charge un caractère
       jz   read2  ;si disponible
       stosb       ;écriture dans le
                   ;buffer
       loop read1  ;tant que < nombre max
read2: mov ax,cx   ;résultat de la boucle
       pop cx      ;nb caractères à
                   ;traiter
       sub cx,ax   ;nb caractères traités
maj_nbcar:
       lds bx,cs:[ddbloc]
                   ;mise à jour
                   ;bloc requête
       mov [bx].nbcar,cx
                   ;car. traités
       jmp ok_exit
                   ;Sortie normale
```

La lecture des caractères dans le buffer interne du DD est un bon exemple d'utilisation d'un buffer circulaire. A noter qu'il faut obligatoirement désactiver les interruptions pendant l'utilisation des pointeurs afin d'éviter toute modification de ces derniers par le traitement de l'interruption série.

```
;—— Récupération de caractère dans le buffer
;—— d'entrée
rcar:  cmp cs:bufin_nb,0  ;caractère en attente ?
       jnz rcar1          ;si non, retour avec
       ret                ;avec flag Z positionné
rcar1:  cli                 ;pas d'interruption
                           ;SVP !
        mov bx,cs:bufin_rd ;récupération
                           ;caractère
        mov al,cs:[bx]     ;grâce au pointeur
                           ;lecture
        inc bx             ;incrément du
                           ;pointeur
        cmp bx,offset bufin_end
                           ;et comparaison
        jne rcar2          ;avec la valeur
                           ;limite
        lea bx,bufin       ;retour du
                           ;pointeur au
                           ;début
rcar2:  mov cs:bufin_rd,bx ;mise à jour de
                           ;la valeur
        dec cs:bufin_nb    ;décrément du
                           ;nombre de car
        sti                ;tout est fini.
        or bx,bx           ;force le flag NZ
        ret
```

L'écriture de données à partir de la fonction MS-DOS int 21h/40h correspond à une demande d'envoi de caractères sur la ligne série. Ceux-ci seront immédiatement acceptés et stockés dans le buffer d'envoi tant qu'il y reste de la place. Il n'y a aucune différence entre les fonctions 08h et 09h, car il est naturellement impossible de vérifier le caractère une fois envoyé.

Le nombre de caractères transférés dépend de la quantité de caractères fournis par le programme appe-

lant, mais aussi de la place disponible dans le buffer circulaire de sortie. Il doit être mis à jour dans le bloc de requête.

```
;—— Point d'entrée de la fonction 08h
;(WRITE)
;—— et 09h (WRITE WITH VERIFY)
write:    push es        ;copie ES:DI dans DS:SI
          pop ds
          mov  si,di
          push cx        ;nb car max à traiter
writ1:    lodsb          ;chargement d'un
                         ;caractère
          call wcar      ;écriture buffer sortie
          jz writ2       ;si place disponible et
          loop writ1     ;tant que < nombre maxi
writ2:    mov ax,cx      ;résultat de la boucle
          pop cx         ;nb car. à traiter
          sub cx,ax      ;nb car. traités
          jmp maj_nbcar ;mise à jour bloc
                         ;requête
```

L'envoi réel des caractères sur la ligne se réalise sous interruption chaque fois que le registre de transmission est vide. Cependant, lorsque cette interruption arrive quand le buffer d'envoi est vide, aucun caractère ne pourra être rechargé dans le registre, et donc plus aucune interruption ne surviendra ultérieurement. Pour éviter ce risque de «désamorçage», le programme doit tester le contenu du buffer et envoyer le caractère directement sur le port sous réserve que celui-ci soit vide.

```
;—— Insertion d'un caractère dans le
;—— buffer de sortie
wcar:     cmp cs:bufout_nb,bufout_lg
                    ;buffer plein ?
          jne wcar1 ;si oui, retour avec
          ret       ;flag Z positionné
wcar1:    cmp cs:bufout_nb,0
                    ;buffer vide ?
          jne wcar2
          push ax   ;sauvegarde du
                    ;caractère
          mov ds,cs:basio
                    ;adresse du port
          add dx,5
          in al,dx  ;lecture du MSR
          test al,00100000b
                    ;Transmit register
                    ;vide?
          pop ax    ;récup du caractère
          jz wcar2  ;non, on met en buffer
          sub dx,5  ;oui, on envoie
          out dx,al ;sur la ligne
          jmp short wcar4
                    ;et c'est déjà fini
wcar2:    cli       ;pas d'interruption SVP!
          mov bx,cs:bufout_wr
                    ;pointeur d'écriture
          mov cs:[bx],al
                    ;insertion caractère
          inc bx    ;incrément du pointeur
          cmp bx,offset bufout_end
                    ; et comparaison avec la
          jne wcar3 ;valeur limite
          lea bx,bufout
                    ;retour au début
wcar3:    mov cs:bufout_wr,bx
                    ;mise à jour
          inc cs:bufout_nb
                    ;incrément  nombre car.
          sti       ;tout est fini.
wcar4:    or bx,bx  ;force le flag NZ
          ret
```

Les fonctions de status permettent de connaître l'état des deux buffers internes au DD. Elles sont grandement facilitées par la gestion du nombre de caractères présents dans chaque buffer. Il faut veiller à ne pas se tromper dans dans le sens du test, le bit «busy» spécifiant à l'application qu'elle doit attendre. Il est donc positionné si aucun caractère ne se trouve disponible en entrée, ou si le buffer de sortie est plein.

```
;—— Point d'entrée de la fonction 06h
;    (INPUT STATUS)
input_status:
      cmp bufin_nb,0 ;caractère en
                     ;attente ?
      jnz inp1
      jmp busy_exit  ;lever bit busy si
                     ;aucun
inp1: jmp ok_exit    ;sortie normale
;—— Point d'entrée de la fonction 0Ah
;    (OUTPUT STATUS)
output_status:
      cmp bufout_nb,bufout_lg
      jne outp1     ;buffer plein ?
      jmp busy_exit ;lever le bit busy si
                    ;plein
outp1:jmp ok_exit   ;sortie normale
```

La lecture non destructrice permet de connaître un caractère en avance sans le sortir du buffer. Elle est réalisée en accédant directement au buffer, sans mettre à jour les pointeurs.

```
;—— Point d'entrée fonction 05h
;      (Lecture non destructrice)
```

```
nd_read:
        cmp bufin_nb,0  ;caractère en
                        ;attente ?
        jne ndrd1       ;si non, sortir en
        jmp busy_exit   ;positionnant bit
                        ;busy
ndrd1:
        mov bx,bufin_rd ;récupération
                        ;caractère
        mov al,[bx]     ;pointeur de
                        ;lecture
        lds bx,[ddbloc] ;stockage du
                        ;caractère
        mov [bx+13],al  ;dans le bloc de
                        ;requête
        jmp ok_exit     ;sortie normale
```

Les fonctions de remise à zéro des buffers en entrée et en sortie sont réalisées simplement en repositionnant les deux pointeurs au début du buffer et en annulant le décompte des caractères.

Tout cela bien entendu en ayant masqué les interruptions pour éviter tout problème d'incohérence si une coupure arrivait au milieu de cette opération.

```
;—— Point d'entrée fonction 07h (RAZ
;    buffer d'entrée)
input_flush:
    cli
    mov bufin_rd,offset bufin   ;RAZ des
    mov bufin_wr,offset bufin   ;pointeurs
    mov bufin_nb,0
    jmp ok_exit                 ;sortie normale
;—— Point d'entrée fonction 0Bh (RAZ
;      buffer de sortie) output_flush:
    cli
    mov bufout_rd,offset bufout ;RAZ des
    mov bufout_wr,offset bufout ;pointeurs
    mov bufout_nb,0
    jmp ok_exit                 ;sortie normale
```

La reconfiguration des paramètres du port série nécessite une programmation des registres du circuit 8250. Cela ne peut se faire à tout moment et dans n'importe quelles conditions. Cette possibilité de reconfiguration implique le support des fonctions facultatives «Open/Close».

La programmation du 8250 et le détournement de l'interruption série ne se fait donc pas une fois pour toutes à l'initialisation du DD, mais à la demande et avec les paramètres par défaut ou ceux transférés par la fonction Write IOCTL.

Lors d'une commande d'ouverture du DD, le premier contrôle consiste à vérifier que le DD n'est pas déjà ouvert et donc utilisé.

Dans la négative, il s'agira de sauvegarder les paramètres de contrôle afin que toute modification ultérieure ne perturbe pas le fonctionnement du DD. Celui-ci devra être fermé puis réouvert pour prendre en compte les nouveaux paramètres.

Le DD doit, enfin, initialiser la partie matérielle (registres du 8250 et du contrôleur d'interruption) avant de remettre à zéro tous les buffers.

PROGRAMMATION

Le port série est géré à partir d'un circuit 8250 ou compatible. Ce circuit présente 9 registres décrits ci-dessous.
Les adresses absolues données sont valables pour le port COM1 (base en 378h).

DATA (base+0 soit 378h pour COM1) : registre permettant de lire un octet reçu ou d'envoyer un octet sur la ligne de transmission.

IER (base+1 soit 379h) Interrupt Enable Register : registre permettant de choisir les conditions amenant la génération d'une interruption.

Les 4 bits de poids faible sont utilisés pour autoriser une interruption :

bit 0 : lors de l'arrivée d'une donnée;

bit 1 : lors du départ effectif sur la ligne de la totalité du caractère. Le registre d'émission est alors vide;

bit 2 : lors d'un problème de ligne (overrun, parité, framing, break);

bit 3 : lors d'un changement d'état des signaux de contrôle modem (RI, DCD, DSR, CTS).

IIR (base+2 soit 37Ah) Interrupt Identification Register : registre permettant, une fois l'interruption générée, d'identifier la condition ayant provoqué cette interruption.

Seuls les 3 bits de poids faible sont utilisés :

bit 0 : indique que le circuit a effectivement généré une interruption;

bits 1 et 2 : codage de la condition :

00 : signaux modem (RI, DCD, DSR, CTS)

01 : registre transmission vide

10 : caractère reçu

11 : signaux de ligne (overrun, parité, framing, break)

LCR (base+3 soit 37Bh) Line Control Register : registre utilisé pour choisir les principaux paramètres de communication (longueur du mot, parité...).

Le format est comme suit :

bit 0 et 1 : longueur en bits du mot transmis ou reçu (00 pour 5 bits, 01=6, 10=7 et 11=8 bits);

bit 2 : nombre de bits de stop générés (0=1 stop, 1=2 stop bits);

bits 3 et 4 : parité (01=impair, 11=pair, x0=pas de parité);

bit 5 : sélection de polarité de la parité (laisser à 0);

bit 6 : contrôle du Break (laisser à 0);

bit 7 : bit DLAB (Divisor Latch Adress Bit).

Quand ce bit est à 1, les adresses 378h et 379h choisissent les registres de sélection de vitesse. Ce bit doit être à 0 en mode normal, pour pouvoir adresser les registres DATA et IER.

MCR (base+4 soit 37Ch) Modem Control Register : permet de contrôler les signaux modem (DTR et RTS).

```
;—— Point d'entrée fonction 0Dh
;    (Ouverture du device)
open  db 0          ;flag d'ouverture
open_dev:
       cmp open,1    ;DD déjà en service ?
       je  open_ko   ;si oui, erreur.
       mov open,1    ;on va l'ouvrir.
       mov ax,comadr ;sauvegarde de
                     ;l'adresse
       mov basio,ax  ;en local
       mov ax,comirq ;calcul de
                     ;l'interruption
       add ax,8      ;utilisée (8 + IRQ)
       mov nivit,ax
       mov cx,comirq ;génération du masque
       mov ax,1      ;pour l'int utilisée
       shl ax,cl       mov msit,al
       mov ax,comirq ;génération du masque
                     ;pour l'acquittement
       add  ax,60h   ;de l'int sur
       mov  ackit,al ;le PIC 8259
       call inio     ;initialisation COM
       jne  open_ko  ;qui peut mal se
                     ;passer
       mov bufin_rd,offset bufin  ;RAZ de
       mov bufin_wr,offset bufin  ;tous les
       mov bufin_nb,0             ;buffers
       mov bufout_rd,offset bufout
       mov bufout_wr,offset bufout
       mov bufout_nb,0
       jmp ok_exit   ;sortie normale
open_ko:
       mov al,0Ch    ;"General failure"
       jmp err_exit  ;et sortie
```

DU PORT SÉRIE

bit 0 : contrôle du signal Data Terminal Ready DTR (1=DTR actif);
bit 1 : contrôle du signal Request To Send RTS (1=RTS actif);
bits 2 et 3 : contrôle des signaux OUT. Ces deux bits doivent être initialisés à 10;
bit 4 : permet le bouclage des sorties sur les entrées afin de tester le circuit;
bits 5 à 7 : inutilisés.

LSR (base+5 soit 37Dh) Line Status Register : fournit des informations sur l'activité de la ligne :
bit 0 : réception d'un caractère;
bit 1 : erreur «overrun». L'avant-dernier caractère reçu n'a pas été lu et a été écrasé par le dernier caractère reçu;
bit 2 : erreur de parité;
bit 3 : erreur de «framing». Le caractère reçu n'avait pas son ou ses bits de stop valides;
bit 4 : détection d'un signal Break (ligne maintenue à 0 pendant un temps supérieur au temps de transmission d'un octet);
bit 5 : registre de transmission vide;
bit 6 : registre de décalage actif;
bit 7 : réservé à 0.

MSR (base+6 soit 37Eh) Modem Status Register : reflète l'état des signaux de gestion modem :
bit 0 : changement d'état sur le signal CTS;
bit 1 : changement d'état sur le signal DSR;
bit 2 : changement d'état sur le signal RI;
bit 3 : changement d'état sur le signal RLSD;
bit 4 : état du signal Clear To Send CTS;
bit 5 : état du signal Data Set Ready DSR;
bit 6 : état du signal Ring Indicator RI;
bit 7 : état du signal Received Line Signal Detect RLSD.

DLL (base+0 soit 378h quand le bit DLAB de LCR est à 1) Divisor Latch Low byte : valeur de la partie basse du diviseur de vitesse.

DLH (base+1 soit 379h quand le bit DLAB de LCR est à 1) Divisor latch High byte : valeur de la partie haute du diviseur de vitesse.

Le circuit 8250 est piloté par une horloge à 1,8432 MHz qui peut être divisée par un nombre variant entre 1 et 65 535. Ce diviseur va être représentatif de la vitesse de transmission ou de réception en bauds.
La formule de calcul est la suivante :
diviseur = 1 843 200 / 16 / vitesse en bauds.
Par exemple, la valeur utilisée pour 9 600 bauds est égale à 12.

Le premier point lors de l'initialisation physique du port série consiste à s'assurer de sa présence en testant les trois bits de poids faible du registre IIR. Ces derniers doivent être tous à zéro. La vitesse, le format des caractères et l'état initial des signaux de contrôles sont ensuite programmés avant d'autoriser les interruptions sur réception de caractères et avec la condition «buffer d'émission vide». Les interruptions générées par le 8250 d'après ces deux conditions sont dirigées vers le PIC 8259 sur lequel il convient aussi de permettre l'interruption. Cela s'effectue en démasquant le bit correspondant au niveau d'interruption sur le port 21h. Il faut, bien entendu, avoir pris auparavant la précaution de remplacer la routine d'interruption courante par la nôtre.

```
;—— Initialisation de la voie série
inio:  mov  dx,basio      ;adresse du port
       add  dx,2          ;pointe sur IIR
       in   al,dx
       test al,0F8h       ;carte présente ?
       jz   inio1         ;si non, retour
       ret                ;avec flag NZ
inio1:cli
       inc  dx            ;pointe sur LCR
       mov  al,80h        ;sélecte DLAB = 1
       out  dx,al         ;accéder à la
                          ;vitesse
       sub  dx,2          ;pointe sur DLM
```

```
        mov  al,byte ptr dvl+1  ;poids fort
        out  dx,al              ;vitesse
        dec  dx             ;pointe sur DLL
mov  al,byte ptr dvl        ;poids faible
        out  dx,al          ;vitesse
        add  dx,3           ;pointe sur LCR
        mov  al,byte ptr fcar
        out  dx,al
        inc  dx             ;pointe sur MCR
        mov  al,00001001b ;OUT2 & DTR hauts
        out  dx,al          ;et RTS bas
        sub  dx,4           ;pointe sur DATA
        in   al,dx          ;vider buffer
                            ;réception
        jmp  short $+2      ;toujours
                            ;attendre un
        jmp  short $+2      ;peu entre deux
        jmp  short $+2      ;opérations
                            ;in/out et
        in   al,dx          ;recommencer
                            ;une fois
        inc  dx             ;pointe sur IER
        mov  al,00000011b ;autoriser IRQ
                            ;réception
        out  dx,al          ;+ transmitter
                            ;empty
        mov  ax,nivit       ;changement de
                            ;vecteur
        mov  ah,35h         ;d'interruption
        push es
        int  21h            ;int MS-DOS
        mov  word ptr old_veccom,bx
        mov  word ptr old_veccom+2,es
        pop  es
        lea  dx,itcom       ;et installation
                            ;du nôtre à sa
        mov  ax,nivit       ;place
        mov  ah,25h
        int  21h            ;int MS-DOS
        mov  dx,21h         ;mot de commande
                            ;PIC 8259
        in   al,dx          ;état masquage
        mov  ah,msit        ;courant
        not  ah             ;pour démasquer
                            ;notre
        and  al,ah          ;interruption
        out  dx,al
        xor  al,al          ;tout est OK
        ret
```

La fermeture du device est simple puisqu'il suffit de désactiver l'interruption série et de remplacer notre routine d'interruption par celle initiale que nous avions remplacée.

```
;—— Point d'entrée fonction 0Eh
;     (Fermeture du device)
close_dev:
        cli             ;pour être tranquille
        mov dx,21h      ;I/O commande 8259
        in  al,dx       ;lire état masquage
        mov ah,msit     ;courant et masquer
                        ;l'interruption
        or  al,ah       ;correspondant au port
        out dx,al
        mov al,byte ptr nivit
                        ;restaure routine IT
        mov ah,25h      ;avec la valeur de
        lds dx,old_veccom   ;l'ancien vecteur
        int 21h
        mov open,0      ;état non ouvert
        jmp ok_exit     ;sortie normale
```

La routine de traitement de l'interruption série est appelée quand un caractère est reçu ou quand le registre de transmission devient vide.

Il faut avant tout sauvegarder les registres utilisés, avant d'observer celui de status pour savoir laquelle des deux conditions citées plus haut a provoqué l'émission de l'interruption. Lors de la réception d'un caractère, celui-ci est inclus dans le buffer de réception et les pointeurs sont mis à jour.

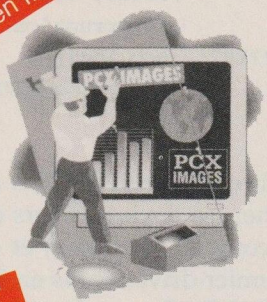
Quand le registre d'émission est vide, un caractère du buffer d'émission est envoyé sur la ligne. L'acquittement de l'interruption sur le 8259 a pour but d'autoriser celui-ci à pouvoir générer une autre interruption.

En effet, afin de hiérarchiser les demandes d'interruptions et pour éviter toute nouvelle entrée, les coupures de priorité égale ou plus faible (ce qui correspond à un numéro d'IRQ plus élevé) ont été désactivées par le 8259 dès la détection de l'interruption série.

```
;—— Traitement des interruptions sur
;       les lignes série
itcom:  push ax             ;sauvegarde de
        push bx             ;tous les
        push dx             ;registres
        push ds             ;utilisés
        push cs
        pop  ds             ;ds=cs
        mov  dx,basio       ;adresse du port
                            ;utilisé
        add  dx,2           ;pointe sur IIR
        in   al,dx          ;lecture du
                            ;status
        cmp al,4            ;caractère reçu?
        je   recep
        cmp  al,2           ;registre vide?
        je   empty
        jmp  acquit         ;Acquittement
```

```
recep:  sub  dx,2          ;pointe sur
                           ;DATA
        in   al,dx         ;récupération
                           ;caractère
        mov  bx,bufin_wr
        mov [bx],al        ;insertion du
                           ;caractère
        inc  bx            ;incrément du
                           ;pointeur et
        cmp  bx,offset bufin_end
        jne  rec1               ;si nécessaire
        mov  bx,offset bufin  ;retour
                           ;début buffer
rec1:   cmp  bx,bufin_rd   ;tour complet?
        je   rec2          ;si oui, car
.                          ;perdu
        inc  bufin_nb      ;si non, mise à
                           ;jour des
        mov  bufin_wr,bx   ;pointeurs
rec2:   jmp  acquit        ;fin réception
empty:  cmp  bufout_nb,0   ;caractères en
                           ;attente ?
        je   acquit
        mov  bx,bufout_rd  ;pointeur de
                           ;lecture
        mov  al,byte ptr[bx]
        sub  dx,2          ;pointe sur
                           ;DATA
        out  dx,al         ;et envoi
        inc  bx            ;incrément
                           ;pointeur
        cmp  bx,offset bufout_end
                           ;comparaison
                           ;avec la
        jne  emp1          ;valeur limite
        lea  bx,bufout     ;pointeur au
                           ;début
emp1:   mov  bufout_rd,bx  ;mise à jour
                           ;valeur
        dec  bufout_nb     ;décrément nb
                           ;car.
acquit: mov  al,ackit      ;acquittement
                           ;de l'int sur
        mov  dx,20h        ;le PIC 8259
```


2 490 FHT
2 953,14 F TTC
manuel en français



RAJOUTEZ LA PUISSANCE DU GRAPHISME A VOS APPLICATIONS

PCX PROGRAMMER'S TOOLKIT 4.0

- Plus de 90 fonctions graphiques
- Possibilité d'affichage de graphismes dans toutes les applications au format PCX
- Fonctions de sauvegarde, d'impression, de manipulation d'images
- Fonctions de capture d'écrans, de gestion de fichiers image
- Possibilité de mélanger textes et graphismes
- Compatible avec 12 compilateurs et 6 langages ; support EMS ; 22 modes vidéo supportés
- Pas de royalties ; version avec source disponnible, version française

1 290 F HT

GX TEXT 2.0
(1529,94 F TTC)

Permet d'écrire des textes tout en étant en mode graphique. Le mélange des images et des textes créés devient très facile. Un éditeur de fontes est inclus pour compléter plus de 70 fontes livrées en standard. Version US.

1 690 F HT

GX EFFECTS
(2003,34 F TTC)

Ajoutez des effets spectaculaires à vos programmes : effet de diaporama, explosion de l'image, affichage de l'image en diagonale... Un sous programme musical permet de mélanger musique et image ! Version US.

1 690 F HT

GX GRAPHICS
(2003,34 TTC)

Une librairie graphique portable qui offre toutes les fonctions graphiques incluses dans le BGI (Borland) ou les MS Librairies mais qui est compatible avec 12 compilateurs.

INNOSOFT

Tél. (1) 40 99 28 00 - Fax : 40 99 28 88

BON DE COMMANDE ET DE DOCUMENTATION à renvoyer à INNOSOFT - 2, rue des Bourets - 92150 Suresnes

I_I Je commande I_I Je désire recevoir des documentation sur vos produits : .....

I_I **PCX PROGRAMMER'S TOOLKIT. 4.0** au prix unitaire de **3024,30 F TTC**
(soit 2953,14 F TTC + 71,16 F TTC de port)
I_I **GX TEXT 2.0** au prix unitaire de **1601,10 F TTC**
(soit 1529,94 F TTC + 71,16 F TTC de port)
I_I **GX EFFECTS** au prix unitaire de **2074,50 F TTC**
(soit 2003,34 F TTC + 71,16 F TTC de port)
I_I **GX GRAPHICS** au prix unitaire de **2074,50 F TTC**
(soit 2003,34 F TTC + 71,16 F TTC de port)

Ci-joint mon règlement :
I_I Chèque I_I Carte Bleue N°... ..... Date d'expiration.....
I_I Contre-remboursement (60 F TTC de frais en sus)

Nom : .....
Société : .....
Adresse :.....
Code postal : I_I_I_I_I_I Ville : .....

```
        out   dx,al
        pop   ds              ;restauration
        pop   dx              ;des registres
        pop   bx
        pop   ax
        iret                  ;retour
                              ;d'interruption
```

La dernière fonction du DD à implémenter représente la première à être appelée : la fonction Init. Elle est située à la fin, car n'étant appelée qu'une seule fois au départ, elle n'a aucune raison de rester résidente. Le DD rend dans le bloc de requête, comme fin du code résident, l'adresse de début de l'Init. Il est possible d'afficher un message pendant le chargement, mais seules les fonctions MS-DOS 01h à 0Ch (Entrées-Sorties caractères), 25h/35h (Récupération-Remplacement des vecteurs d'interruptions) et 30h (Numéro de version) peuvent être appelées. Cela est dû au fait que le chargement du DD intervient avant celui du MS-DOS en entier.

```
;—— Point d'entrée fonction 00h (INIT)
init: lds   bx,cs:[ddbloc]     ;adresse bloc
                               ;de requête
      mov   [bx+14],offset init ;adresse
                               ;de fin du
                               ;DD
      mov   [bx+16],cs         ;résident
      xor   ax,ax
      mov   [bx+18],ax         ;RAZ
      mov   [bx+20],ax         ;nécessaires
      mov   [bx+13],al
      push cs                  ;affichage du
      pop   ds                 ;message
      lea   dx,msg
      mov   ah,9
      int   21h                ;int MS-DOS
      jmp   ok_exit            ;sortie normale
msg   db    13,10,"DDCOM Device Driver"
      db    13,10,"Version JFDa 1.00"
      db    13,10,10,'$'
      END
```

Un device driver est une image strictement binaire qui est chargée en mémoire sans PSP. Il commence donc à l'adresse zéro du segment de chargement et non pas à l'adresse 100h comme un fichier .COM. Ainsi s'explique la directive ORG placée au début du programme. Le DD exécutable s'obtient par les instructions suivantes :

```
MASM DDCOM;
LINK DDCOM;
EXE2BIN DDCOM.EXE DDCOM.SYS
```

A noter que la procédure d'édition de liens (Link) génère deux messages d'avertissement «WARNINGS» car le programme ne contient ni adresse de début ni segment de pile. Ces avertissements sont normaux et sans effet sur le fonctionnement correct du DD. Une fois chargé à partir du Config.SYS, l'accès au DD se fait simplement en ouvrant un fichier du nom d'ASYN et en y écrivant les données que l'on souhaite envoyer, tandis que la lecture de ce fichier donne les caractères reçus.

La ligne est initialisée par défaut en 2 400 bauds, mais peut être changée par l'accès IOCTL (int 21h / 4403h). La modification n'est toutefois opérationnelle qu'après une fermeture et une réouverture du «fichier» Asyn.

Anatomie d'un device driver caractère (DDC)

L'en-tête («Header») doit être placé au tout début du code. Il permet d'identifier le driver en lui donnant son nom, en définissant ses attributs et ses points d'entrée. La structure suivante décrit cet en-tête :

```
DEVICEHEADER  STRUC
dhLink        dd ?             ;lien vers le driver
                               ;suivant dhAttributes
              dw ?             ;attributs du driver
                               ;dhStrategy
              dw ?             ;offset routine
                               ;stratégie dhInterrupt
              dw ?             ;offset routine
                               ;interruption
dhNameOrUnits db '????????'    ;nom logique
                               ;du device DEVICEHEADER
ENDS
```

Le champ dhLink doit contenir 0FFFFh. Il est utilisé par MS-DOS pour chaîner les device drivers entre eux, en indiquant l'adresse complète (segment et offset) du driver précédent. Le premier driver (mais en fait le dernier de la liste puisque MS-DOS consulte celle-ci à partir de la fin) contient 0FFFFh, indiquant l'arrêt de la liste. Le champ dhAttributes permet d'identifier le type de device driver et de donner des informations sur ses possibilités. Chaque bit (quand il est mis à 1) s'utilise à la manière d'un flag. Les bits réservés doivent être laissés à 0.

Bit	Signification
0	Le DD est utilisé pour l'entrée standard (STDIN).
1	Le DD est utilisé pour la sortie standard (STDOUT).
2	Le DD est le device NUL. A noter que ce bit ne doit jamais être utilisé car le device NUL résident ne peut être remplacé.
3	Le DD est de type «CLOCK» (horloge).
4	Le DD supporte les sorties rapides par

	l'intermédiaire de l'int 29h
5	Réservé
6	Le DD supporte la fonction 13h «Generic IOCTL».
7	Le DD supporte la fonction 19h «IOCTL Query».
8-10	Réservés
11	Le DD supporte les fonctions 0Dh et 0Eh «Open Device» et «Close Device».
12	Réservé
13	Le DD supporte la fonction 10h «Output Until Busy»
14	Le DD supporte les fonctions 03h «IOCTL Read» et 0Ch «IOCTL Write»
15	Indique si le DD est de type caractère ou bloc. Le bit doit être mis à un pour un DD de type caractère (DDC).

Les champs dhStrategy et dhInterrupt contiennent les points d'entrée des deux routines permettant d'appeler le DD. Seul l'offset se trouve spécifié, le segment devant être le même que celui de l'en-tête. Le champ dhNameOrUnits est utilisé dans le cas d'un DDC pour spécifier le nom du device logique sous la forme de 8 caractères. Le nom doit être complété avec des espaces et ne pas contenir le caractère «:».

Appel d'un device driver

Un DD est toujours appelé par MS-DOS en deux temps :

- ❏ Appel de la routine de stratégie par un Call Far, avec les registres ES:BX pointant sur l'adresse d'un bloc de requête. Le DD doit sauvegarder cette adresse et rendre immédiatement la main au MS-DOS.
- ❏ Appel de la routine d'interruption par un Call Far. Laquelle a pour charge de traiter complètement la requête spécifiée lors de l'appel à la routine de stratégie précédente.

Le bloc de requête est défini comme suit :

```
REQUESTHEADER STRUC
rhLength    db  ?          ;longueur du bloc
                           ;en octets rhUnit
            db  ?          ;non utilisé pour un
                           ;DDC
rhFunction  db  ?          ;numéro de fonction
rhStatus    dw  ?          ;status (flags de
                           ;résultat)
rhReserved db 8 dup (?)    ;réservé
REQUESTHEADER ENDS
```

Cette structure représente le début invariable et minimal (13 octets) d'un bloc de requête. Suivant la fonction demandée, celui-ci pourra être complété par d'autres paramètres mis directement à sa suite. Le champ rhLength spécifie la longueur totale en octets du bloc. Le champ rhFunction indique la fonction à réaliser, tandis que le champ rhStatus permet au DD de communiquer un status résultat au programme appelant. Trois bits du status sont principalement utilisés :

Le bit 9 indique que le device est occupé («busy»). Ce bit est positionné notamment sur un appel des fonctions 06h «Input Status» et 0Ah «Output Status».

Le bit 8 signale que l'opération a été effectuée. Le DD doit mettre ce bit à un quand il rend la main au MS-DOS.

Le bit 15 révèle une erreur. Lorsqu'il est positionné, les 8 bits 0-7 du mot status indiquent le type de l'erreur.

Les 16 premières valeurs sont prédéfinies. La valeur 3, par exemple, sert à indiquer une commande inconnue tandis que 0Ch est utilisé pour «General failure».

Fonctions d'un device driver caractère

• Fonction 00h (INIT) : cette fonction permet d'initialiser le DD. Elle n'est appelée qu'une seule fois au moment où le DD est chargé par MS-DOS lors de l'analyse du fichier Config.SYS.

```
INITREQUEST  STRUCT
irRequest    db  13 dup (?) ;bloc
                           ;invariant
irUNITS      db  ?         ;non utilisé
                           ;pour un DDC
irEndAdress  dd  ?         ;ENTREE:fin
                           ;de la mémoire
                           ;disponible
                           ;pour le
                           ;driver
                           ;SORTIE:fin du
                           ;code résident
irParamAdress dd ?         ;ENTREE:
                           ;adresse de la
                           ;ligne dans le
                           ;CONFIG.SYS
                           ;SORTIE:0 pour
                           ;un DDC
irDriveNumber db ?         ;non utilisé
                           ;pour un DDC
irMessageFlag dw ?         ;Autorise MS-
                           ;DOS à
                           ;afficher un
                           ;message
                           ;d'erreur
INITREQUEST ENDS
```

• Fonctions 04h (Read), 08h (Write) et 09h (Write With Verify): ces fonctions sont utilisées pour lire et écrire les

caractères sur le périphérique («device») géré par le DD. Le programme appelant spécifie l'adresse de son buffer et le nombre de caractères à traiter.

```
READWRITEREQUEST  STRUC
rwrRequest    db 13dup (?);bloc invariant
rwrMediaID    db ?        ;non utilisé
                          ;pour un DDC
rwrBuffer     dd ?        ;adresse du
                          ;buffer
rwrBytesSec   dw ?        ;ENTREE: nombre
                          ;de caractères
                          ;à lire ou
                          ;écrire
                          ;SORTIE: nombre
                          ;de caractères
                          ;lus ou écrits
rwrStartSec   dw ?        ;non utilisé
                          ;pour un DDC
rwrVolumeID   dd ?        ;non utilisé
                          ;pour un DDC
rwrHStartSec  dd ?        ;non utilisé
                          ;pour un DDC
READWRITEREQUEST ENDS
```

• Fonction 05h (Non-Destructive Read) : cette fonction permet de récupérer le premier caractère du buffer d'entrée sans le faire disparaître de ce dernier. Toute opération de lecture ultérieure retournera donc le même caractère. Si le buffer ne comporte pas de caractère, le DD doit positionner le bit 9 «busy» dans le mot de status.

```
NDREADREQUEST  STRUC
nrrRequest   db 13dup (?) ;bloc invariant
nrrChar      db ?         ;caractère lu
NDREADREQUEST  ENDS
```

• Fonctions 06h (Input Status) et 0Ah (Output Status) : ces fonctions s'utilisent pour connaître l'état des buffers d'entrée ou de sortie du DD. Si aucun caractère n'est en attente dans le buffer d'entrée (Input Status), ou si des caractères sont encore présents dans le buffer de sortie (Output Status), le DD doit positionner le bit 9 «busy» dans le mot de status. Le bloc de requête pour ces fonctions ne requiert aucune extension et se limite donc au bloc minimal de 13 octets.

• Fonctions 07h (Input Flush) et 0Bh (Output Flush) : ces fonctions permettent de vider les buffers d'entrée ou de sortie du DD. Le bloc de requête pour ces fonctions ne requiert aucune extension et se limite donc au bloc minimal de 13 octets.

• Fonctions 0Dh (Open Device) et 0Eh (Close Device) : ces deux fonctions sont appelées par MS-DOS lorsqu'une application ouvre ou ferme un périphérique («device»). Un DDC peut implémenter ces fonctions pour initialiser un périphérique lors de son ouverture. Le bloc de requête pour ces fonctions ne réclame aucune extension et se limite donc au bloc minimal de 13 octets.

• Fonction 10h (Output Until Busy) : cette fonction permet de transférer des données dans le buffer de sortie du DD.

Celui-ci doit rendre immédiatement la main au programme appelant en spécifiant le nombre de caractères effectivement pris en compte.

```
OUTPUTREQUEST  STRUC
orRequest  db 13dup (?)  ;bloc invariant
orData     db ?          ;non utilisé
orBuffer   dd ?          ;adresse du
                         ;buffer orBytes
           dw ?          ;ENTREE: nombre
                         ;de caractères à
                         ;envoyer
                         ;SORTIE: nombre
                         ;de caractères
                         ;effectivement
                         ;pris en compte
OUTPUTREQUEST  ENDS
```

• Fonctions 03h (IOCTL Read) et 0Ch (IOCTL Write) : ces deux fonctions permettent de transférer des données entre le DD et un programme appelant. Elles ne sont appelées qu'une fois que le bit 14 des attributs du DD se trouve positionné. Le format des données échangées est spécifique au DD.

```
IOCTLRREQUEST STRUCT
irwrRequest db 13dup (?)  ;bloc invariant
irwrData    db  ?         ;non utilisé
irwrBuffer  dd  ?         ;adresse du
                          ;buffer
irwrBytes   dw  ?         ;ENTREE:nombre
                          ;de caractères
                          ;à transférer
                          ;SORTIE: nombre
                          ;de caractères
                          ;transférés
IOCTLRREQUEST ENDS
```

• Fonctions 19h (IOCTL Query) et 13h (Generic IOCTL) : la fonction 19h (IOCTL Query) permet de savoir si le DD supporte une sous-fonction particulière de la fonction 13h (Generic IOCTL).

Dans le cas où cette sous-fonction n'est pas supportée, le DD doit rendre une erreur (bits 8 et 15 positionnés)

avec le code 03h (Unknown function) dans les 8 bits de poids faible du status.

```
IOCTLREQUEST  STRUC
giRequest    db 13 dup(?) ;bloc invariant
giCategory   db ?          ;catégorie du
                           ;device
giMinorCode db ?           ;numéro de la
                           ;sous-fonction
giReserved   dd ?          ;non utilisé
giIOCTLData dd ?           ;adresse des
                           ;données
                           ;spécifiques à
                           ;la sous-fonction
IOCTLREQUEST  ENDS
```

La catégorie du DD peut prendre plusieurs valeurs (01h pour série, 03h pour console, 05h pour parallèle et 08h pour disque).

Les sous-fonctions utilisées par un DDC peuvent être les suivantes :

45h Spécifie le nombre d'itérations effectuées par le DD avant qu'il ne soit «busy».

65h Rend le nombre d'itérations utilisées par le DD.

4Ah Sélection d'une page de code.

4Ch Début de préparation d'une page de code.

4Dh Fin de préparation d'une page de code.

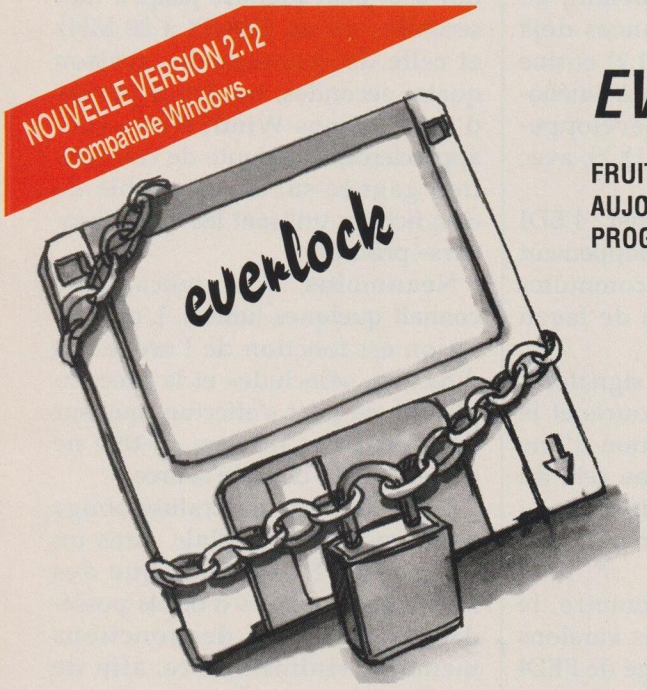
6Ah Rend la page de code courante.

6Bh Rend la liste des pages de code supportées.

Toutes ces sous-fonctions sont accessibles par un programme MS-DOS et par l'intermédiaire des fonctions 44xx de l'int 21h.

BIBLIOGRAPHIE

- Microsoft Programmer's Library CD-ROM. Microsoft press book.
- MS-DOS Programmer's Reference. Microsoft press book.
- The MS-DOS Encyclopedia (Ray Duncan). Microsoft press book.
- Intel 8250 Data Sheet.



EVERLOCK, L'ANTIVOL LOGICIEL

FRUIT DE NOMBREUSES ANNÉES D'EXPÉRIENCE, EVERLOCK S'IMPOSE AUJOURD'HUI COMME LE LOGICIEL DE CHOIX POUR PROTÉGER TOUS VOS PROGRAMMES DOS ET WINDOWS :

- **Très haute protection contre le désassemblage, la copie octet par octet, reinstall, copyII PC, copywrite, etc...**
- **Encryption unique de vos fichiers**
- **Paramétrage très souple avec compteurs du nombre de copies, d'utilisateurs, de jours...**
- **Mises à jour à distance de vos logiciels**
- **Transparence totale pour l'utilisateur en mono-poste ou réseau**
- **Installation rapide et conviviale sans disquette spéciale**
- **Très faible coût par copie**

2 950 F HT *
(3 498,70 F TTC)
POUR 120 COPIES

Pour commander une disquette de démonstration appelez ou écrivez à : INNOSOFT

Tél. (1) 40 99 28 00 . 2, rue des Bourets - 92150 Suresnes - Fax : 40 99 28 88

* 2 950 F HT (3 498,70 F TTC) pour 120 utilisations . 8 950 F HT (10 614,70 F TTC) pour la version illimitée. Documentation en français.

BON DE COMMANDE - A renvoyer à INNOSOFT 2, rue des Bourets - 92150 Suresnes

Veuillez trouver ci-joint un chèque de 50 FTTC et me faire parvenir la disquette de démonstration EVERLOCK ainsi que sa documentation.

Nom : ..... Société : .....

Adresse : ..... Code postal : I_I_I_I_I_I Ville : .....

Service lecteurs n° 12 (dernière page)

LANGAGE

C++ OU PASCAL

Qu'importe le langage pourvu qu'il soit objet !

ALAIN DORSEUIL

La programmation par objets se prête bien à la mise en oeuvre d'applications dotées d'interfaces «utilisateur» évoluées. Mais quels sont les outils à la disposition du programmeur dans ce domaine?

Si en matière d'interfaces «utilisateur» une large majorité se dégage en faveur de Windows, quelles alternatives s'offrent, là encore, aux développeurs?

C'est du côté de chez Borland que l'on décèle quelques éléments de réponse à ces questions, grâce en partie à deux nouveaux produits de la firme : le Borland C++ v2.0 qui semble être en mesure de remplacer le couple C-Microsoft-SDK pour la mise en œuvre d'applications Windows; le Turbo Pascal orienté objet v6.0 qui, avec son environnement «objet» Turbo vision, se présente comme le futur standard de programmation pascal en environnement DOS.

Le Borland C++ version 2.0

Si le Turbo C++ constitue le produit de base de la gamme Borland pour s'initier au langage C++ et à la programmation par objets sous MS-DOS, le Borland C++ se présente davantage comme un outil de développement professionnel visant à concurrencer le couple C-Microsoft-SDK dans la mise en œuvre d'applications Windows.

Commençons donc par un inventaire sommaire du contenu du pack de développement : un compilateur C++ (au standard AT&T 2.0), un compilateur C (à la norme C-Ansi), un Assembleur modifié pour fonctionner en mode protégé à partir de Windows (Turbo Assembleur v2.5), un éditeur de liens capable de prendre en compte les spécificités de Windows (Tlink), un débogueur DOS et Windows (Turbo Debugger v2.5), un éditeur de ressources (Whitewater Resource Toolkit), un analyseur de performances déjà ancien (Turbo Profiler v1.1) et une foule d'utilitaires destinés à améliorer les possibilités de développement en environnement DOS, avec ou sans Windows.

N'oublions pas l'essentiel : l'EDI (Environnement de développement intégré) qui permet de commuter d'un utilitaire à l'autre de façon conviviale et homogène.

A propos de cet EDI, signalons, outre la gestion de la souris et le multifenêtrage, l'apparition d'une fonction Défaire-Refaire au sein du menu et la possibilité d'effectuer du couper-copier-coller à partir du système d'aide.

Côté optimisation mémoire, le Borland C++ intègre des versions mode réel et mode protégé de l'EDI et des compilateurs C/C++, les versions en mode protégé étant exploitables sous Windows moyennant modification de l'autoexec.bat.

Côté performances, en compilation, le Borland C++ s'avère un peu plus rapide que ses concurrents grâce en partie au mécanisme de précompilation des fichiers en-tête.

Lors de la mise en œuvre d'une application faisant intervenir plusieurs fichiers en compilation, le Borland C++ analyse une fois pour toutes les fichiers «header» redondants et génère sur disque une table de symboles commune à tous les fichiers.

Sachant que, selon la mémoire disponible, l'analyse de «windows.h» peut prendre jusqu'à huit secondes sur un i386-SX à 20 MHz et celle de «iostream.h» environ quatre secondes, les développeurs d'applications Windows en C++ apprécieront la minute de compilation gagnée sur un ensemble de cinq fichiers utilisant les deux «headers» précités.

Néanmoins, ce mécanisme connaît quelques limites. L'optimisation est fonction de l'ordre des directives «#include» et la précompilation ne peut s'effectuer que sur des véritables fichiers en-tête ne contenant pas de code source.

Cette dernière contrainte oblige notamment à ne définir, dans un fichier en-tête en C++, que des interfaces de classes d'objets possédant uniquement des fonctions membres «inline», et ce, afin de profiter du gain de temps, aussi minime soit-il, occasionné par la précompilation.

Hormis ces questions de performances et d'optimisation de mémoire, dont l'importance reste relative, le Borland C++ se démarque radicalement, surtout en

Le langage C est le plus utilisé pour le développement d'applications par le biais de la programmation par objets. Mais il ne peut prétendre à l'exclusivité. Le pascal constitue, entre autres, une excellente solution. Borland, avec le Turbo Pascal 6.0 et le C++ 2.0, propose ainsi deux langages orientés objet dotés d'environnements de développement très complets. Lequel choisir et pour quoi faire? Existe-t-il par ailleurs des problèmes de portage des sources écrites avec les versions précédentes des langages du même éditeur? Des réponses peuvent être proposées. Les voici.

phase de débogage, du Turbo C++.

Turbo Debugger n'est plus à présenter aux utilisateurs des chaînes de développement de la famille Borland. Du Turbo Assembleur au Borland C++ en passant par le Turbo C et le Turbo Pascal, cet outil de mise au point convient depuis fort longtemps déjà aux programmeurs de par sa puissance, sa convivialité et sa souplesse.

La version 2.0 avait, dès sa sortie, fait grand bruit en intégrant des possibilités de débogage pour les résidents (TSR) et les pilotes de périphérique (driver). Outre les détails liés au confort de l'utilisateur ou à l'optimisation en vitesse, la version 2.5 apporte deux nouveautés essentielles : la possibilité de déboguer les applications Windows ainsi qu'une meilleure gestion des spécificités de la programmation par objets.

Le débogage des objets

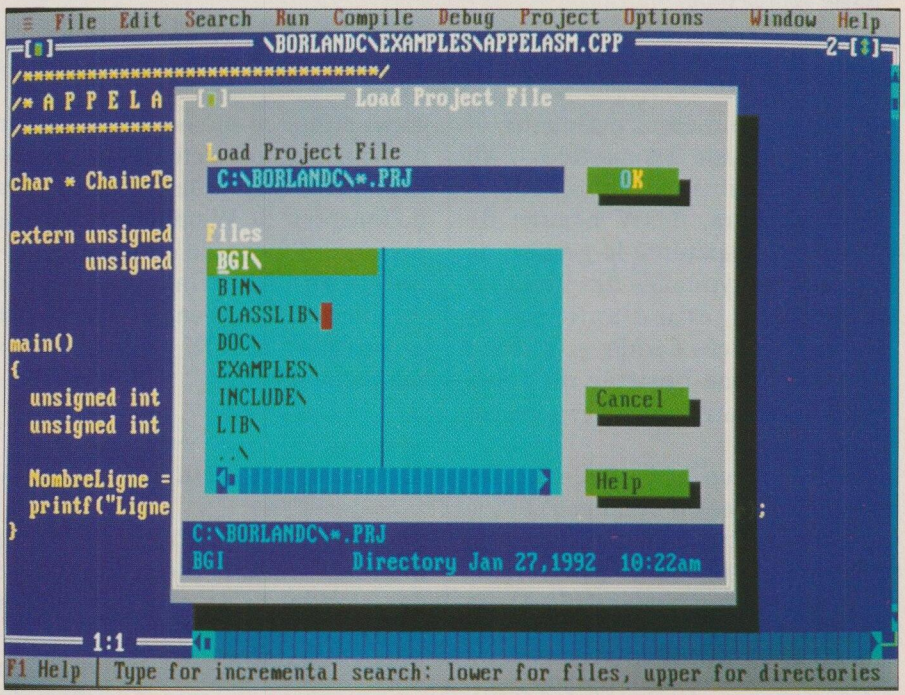
La prise en compte de la programmation par objets sous Turbo Debugger date de la version 2.0. Parmi les nouvelles fonctionnalités, citons les possibilités de visualisation de la hiérarchie des classes d'objets (en héritage simple ou multiple), la visualisation de la structure (attributs + méthodes) d'une classe et même d'une instance de classe.

Cette dernière possibilité s'avère intéressante pour le contrôle de l'implémentation du polymorphisme, compte tenu du fait que les informations présentées incluent les adresses de début de code des méthodes associées à l'objet. En particulier, on peut disposer de l'adresse de base de la table de méthodes virtuelles (Virtual Method Table).

Un débogueur pour applications Windows

La grande nouveauté en 2.5 reste essentiellement le débogage des applications Windows (version 3.x) grâce à TDW (Turbo Debug for Windows) inclus dans le pack Borland C++. Sur un même moniteur, le programmeur dispose d'un écran pour TDW (en mode Texte) et d'un autre pour l'application sous Windows (mode graphique CGA, EGA, VGA ou Hercules monochrome). Au chapitre des restrictions, notons au passage que TDW ne supporte pas (encore?) le mode SuperVGA et que pour l'utiliser il faut au minimum une machine 386 avec 1 Mo de mémoire vive.

Quoi qu'il en soit, TDW permet d'accéder aux spécificités des applications Windows telles que : les messages échangés entre les fenêtres de l'application, le contenu



Le C++ bénéficie de l'interface «à la Borland» avec tous les outils indispensables à la programmation.

LE TPOO, ENTRE LE C++ ET LE PASCAL

Le Turbo Pascal orienté objet hérite d'une philosophie Object Pascal (Apple) et C++. Sans prétendre égaler la puissance et la complexité de ce dernier, il implémente tout de même un nombre suffisant de concepts «objets» pour mériter pleinement son appellation.

• TPOO 6.0 supporte, comme le C++, la notion de méthodes et de variables privées (attribut «private»). Néanmoins, TPOO n'implémente pas de champs protégés (protected) uniquement vus par les classes dérivées.

•En TPOO, les implémentations des méthodes (procédures) sont obligatoirement séparées de la définition de l'interface de l'objet (placée pour commodité dans la partie visible d'une unité).
En C++, l'attribut «inline» permet de définir des macrofonctions au sein même de la classe. Quant à l'identificateur «Self», il repère l'objet receveur (rôle du «this» en C++).

• Le TPOO définit deux mots-clés «constructor» et «destructor» pour indentifier les constructeurs et destructeurs d'objets (en C++, il s'agit de méthodes dont le nom est celui de la classe elle-même). Si en C++, l'appel d'un constructeur est implicite à la déclaration d'une instance, en TPOO, il faut faire un appel explicite au constructeur (même contrainte pour les destructeurs). En outre, il peut y avoir plusieurs destructeurs pour un objet en TPOO, alors qu'en C++, un destructeur de classe ne peut être surchargé.

• La surcharge des opérateurs (+,-,=, etc.), courante en C++, est impossible en TPOO (il convient de définir des fonctions telles que «additionner», «soustraire», etc.).

• La gestion des erreurs d'allocation dynamique au sein d'un constructeur peut être assurée par une procédure standard FAIL en TPOO. Tout reste à faire en C++.

• Le TPOO n'implémente pas de généricité «directe» (paramétrage de types) contrairement au C++ (grâce au mot-clé «template» en C++ AT&T 3.0 uniquement).

• Le polymorphisme est implémenté en TPOO, tout comme en C++, par l'attribut «virtual». Le TPOO fournit en outre un attribut «abstract» pour la spécification d'objets abstraits, alors qu'en C++, il convient de spécifier une fonction virtuelle «pure» (=0).

• Le TPOO n'implémente pas l'héritage multiple contrairement au C++.

des zones de mémoire globales et locales ou la liste des modules (EXE et DLL) constituants l'application.

Par ailleurs, TDW assure de manière transparente la gestion de la table des symboles des librairies dynamiques, afin d'autoriser le débogage de ces dernières, sachant qu'il est même possible de poser des points d'arrêt au niveau de ces DLL.

Enfin, en ce qui concerne les aspects de mise au point et TDW, le développeur appréciera, dans la documentation, l'exemple détaillé du débogage d'une application Windows, fournie avec son fichier «projet».

Néanmoins, d'un point de vue conception, ce même développeur ne pourra que regretter l'absence de documentation sur les bibliothèques de fonctions de Windows, malgré la présence d'une aide en ligne conséquente sur l'API Windows. Pendant combien de temps encore faudra-t-il se procurer le SDK pour la «doc» uniquement?

Avec ou sans le SDK?

Le Software Development Kit de Microsoft faisait jusqu'alors office de «standard» de programmation des applications Windows (en langage C). Le Borland C++, avec son environnement intégré, risque de renverser la tendance, et cela pour trois raisons : il reste compatible avec le SDK (il intègre d'ailleurs deux outils standards du SDK de Microsoft : le compilateur de ressources RC et l'API, interface de programmation d'applications Windows); il facilite le lien entre diverses applications Windows existantes grâce à deux utilitaires très pratiques lors de l'utilisation de DLL mettant en œuvre des classes C++ : IMPDEF (création de définitions exportables) et IMPLIB (création de librairies d'importation); il propose un utilitaire plus convivial que ceux du SDK pour la création et la modification des ressources : le Whitewater Resource Toolkit (WRT).

Un éditeur de ressources très «visuel»

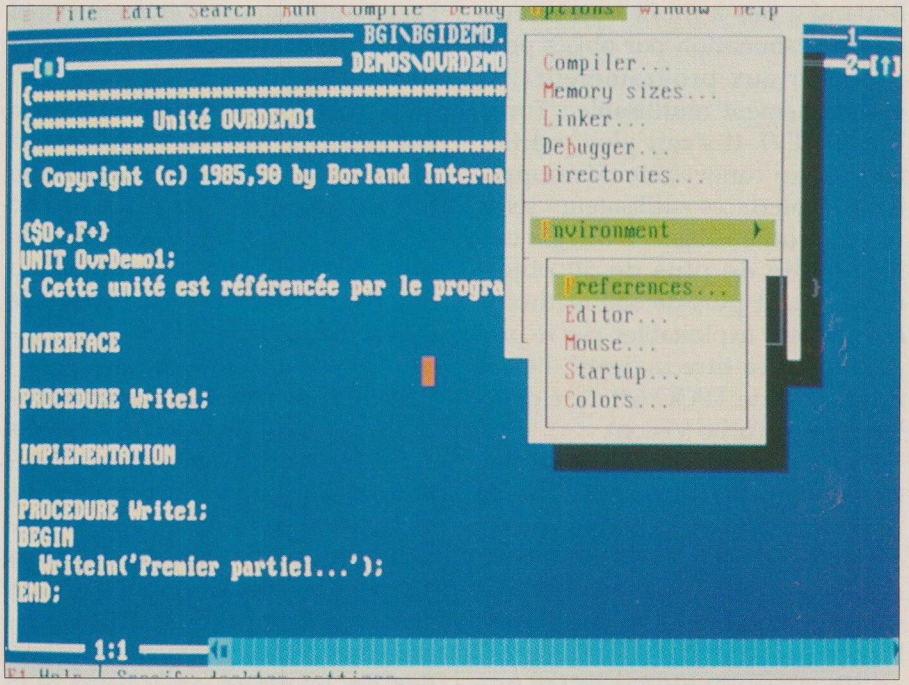
Le Whitewater Resource Toolkit est un gestionnaire de ressources fonctionnant sous Windows 3.0 et constitué de divers éditeurs : boîtes de dialogue, graphiques en mode point, menus, icônes, curseurs, chaînes de caractères et raccourcis-clavier. Là où, pour d'autres environnements de programmation sous Windows, il fallait recourir à des fichiers script de type «.RC» et «.DLG», saisis avec un éditeur ASCII, le toolkit du Borland C++ propose une construction visuelle des ressources et génère automatiquement, au passage, les précédents scripts. Dès lors, on peut s'interroger sur l'utilité du compilateur de ressources de Microsoft concernant les modifications ayant trait uniquement à l'aspect présentation (look) d'une application. Cela d'autant plus que WRT peut directement agir sur des fichiers «.RES» déjà compilés ou encore effectuer, pour duplication ou modification, une extraction des ressources d'une application existante (à partir du programme exécutable).

Considérons tout d'abord la com-

patibilité «interne», à savoir celle qui concerne les produits Borland mêmes, et plus précisément le Turbo C en version 2.0. Le portage des applications du Turbo C 2.0 vers le Borland C++ 2.0 impose malheureusement une recompilation complète car les fichiers «.OBJ» et «.LIB» du Turbo C posent des problèmes de résolution de références à TLINK sous Borland C++. Côté compatibilité «externe» (autres produits que ceux de Borland), la recompilation avec le SDK d'une application Windows, initialement développée sous Microsoft C (6.0), ne pose pas d'autres problèmes que la modification de quelques variables d'environnement, d'autant que le Borland C++ supporte la définition d'un symbole «__MSC» qui autorise le remplacement du nom d'une fonction de librairie définie sous Microsoft C par son équivalent sous Borland C++. Néanmoins, pour de gros projets, soulignons la difficulté de transcodage des informations issues de l'utilitaire MAKE de Microsoft (en version 5.1) vers le MAKE du Borland C++ 2.0. Par contre, pour NMAKE du même Microsoft, Borland fournit une table de transcodage ainsi qu'un utilitaire de conversion en fichier «Project» directement exploitable par l'environnement de programmation. Cela étant, la réalisation d'un portage nécessite une bonne documentation. Là encore, Borland met tous les atouts de son côté pour «récupérer» le maximum de développeurs qui disposent d'une description détaillée de la bibliothèque du Borland C++ où sont précisés pour chaque fonction les cas de portabilité (exclusivité Borland C++, MS-DOS, PC, incompatibilité Windows, conformité C-Ansi, Unix, etc.), des tables de correspondance entre les options des compilateurs, éditeurs de liens de Borland (BCC, TLINK) et ceux de Microsoft (LC, LINK). Conforme au standard AT&T 2.0, le Borland C++ constitue pour l'instant l'unique implémentation pour la mise en œuvre d'applications Windows. Certains regretteront que ce produit ne soit pas entièrement Windows (bien que fonctionnant avec ce dernier), à l'inverse des nouveaux environnements de programmation tels que Visual Basic, ToolBook 1.5, Enfin/2, Actor ou Turbo Pascal for Windows qui sont a priori dédiés à la réalisation d'applications Windows de moindre complexité.

Le Turbo Pascal orienté objet version 6.0

A l'instar du Borland C++, le Turbo Pascal orienté objet (TPOO) bénéficie d'un nouvel environnement de programmation sophistiqué. Même si ce dernier s'inspire fortement des précédentes versions du Turbo Pascal, les améliorations sont loin d'être négligeables : gestion de la souris, édition simultanée de plusieurs fichiers dans différentes fenêtres reconfigurables, mise à disposition d'un presse-papiers (copier-couper-coller) et possibilité de créer ses propres macrocommandes d'édition. Du point de vue de l'interfaçage avec l'assembleur et de la mise au point, citons encore la gestion automatique de mémoire EMS, l'intégration d'un assembleur «inline», la possibilité de poser des points d'arrêt conditionnels au niveau source ainsi que la visualisation des registres du CPU. Notons, pour terminer, la présence d'une aide en ligne de plus de 1 500 écrans, à partir de laquelle il est possible d'effectuer du copier-coller afin de récupérer les exemples didactiques... Parallèlement à l'environnement, le langage lui-même n'a que peu évolué. Si la version 4.0 avait apporté la notion d'unité de compilation, la version 5.5 celles d'objets, d'héritage et de polymorphisme, la version 6.0 ne fait que renforcer le principe d'encapsulation afin de mettre en rapport de manière plus cohérente la nature privée des objets avec les capacités d'abstraction de données des unités. Le TPOO fournit, en effet, un attribut «private» pour la protection des accès sur les champs et méthodes dans une



Le Turbo Pascal est devenu un véritable standard en programmation pascal sur PC.

déclaration d'objet. Indispensable pour une application correcte des concepts de programmation par objets, cet attribut avait cruellement fait défaut dans la version 5.5; le programmeur se trouvant alors dans l'incapacité de protéger les accès sur un champ «donnée» interne à un objet ou encore de contrôler les appels à une méthode liée à l'implémentation (principe de masquage de l'information).

Suite à l'apparition de cette notion de champs et méthodes privées, l'unité OBJECTS de Turbo Pascal 6.0 a été quelque peu modifiée par rapport à celle de la 5.5 (prise en compte de l'attribut «private» pour les objets de type collection et flux).

Si toutes les unités et tous les programmes de la version 5.5 fonctionnent avec la version 6.0 (après recompilation), le seul handicap (de taille) est que l'unité OBJECTS de la version 5.5 ne fonctionne plus avec Turbo Vision, pièce maîtresse de cette nouvelle mouture de TPOO.

Turbo vision : le kit de développement «objet»

Borland a mis à profit les concepts de programmation par objets pour proposer aux programmeurs un environnement réutilisable : Turbo Vision (TV). Il s'agit plus précisément d'un canevas de développement d'interfaces «utilisateur» selon une approche «objet». Il intègre une hiérarchie de plus de 50 objets représentant plus de 350 méthodes directement exploitables par le programmeur (à titre indicatif, l'environnement de TPOO 6.0 a été développé avec Turbo Vision).

L'un des intérêts de Turbo Vision, et non des moindres, est donc de familiariser le programmeur avec les concepts «objets» dans un environnement de programmation par événements, et ce, dans la lignée de toutes les interfaces graphiques «utilisateur» évoluées telles que Presentation Manager sous OS/2, X-Window/Motif en environnement Unix et, bien entendu, Windows sous MS-DOS.

Un programme écrit avec Turbo Vision hérite (au sens «objet») de nombreuses fonctionnalités telles que la gestion de la souris et le contrôle de l'affichage (boîtes de dialogue, menus déroulants, fenêtres, barres de défilement, cases à cocher, boutons radio, etc.). Plusieurs unités prêtes à l'emploi sont par ailleurs à la disposition du programmeur (système d'aide, calculatrice, agenda, etc.) afin de faciliter la manipulation des indispensables accessoires de bureau au sein des programmes applicatifs.

Le composant principal d'une application Turbo Vision est un élément visuel surnommé un «elvis». Il peut s'agir d'une fenêtre, d'un menu, d'une boîte de dialogue ou bien d'une simple chaîne de caractères. Tout «elvis» possède un même objet ancêtre de type TObject (*voir figure 2*) et doit être capable de s'afficher (par surcharge d'une méthode «draw») et de réagir à un ou à plusieurs événements. Ainsi, le rôle de l'application (objet de type TApplication) est de transmettre l'action utilisateur à l'élément visuel concerné, ce dernier réagissant en conséquence.

Le principe de programmation par événements peut s'appliquer à toute ressource faisant partie de la hiérarchie des objets de Turbo Vision. Il serait même possible de procéder à une dérivation (relation d'héritage) afin de particulariser le mécanisme de gestion des ressources (sait-on jamais!).

Soulignons cependant que l'application à un tel niveau des concepts «objets», associée à une programmation par événements incontournable, ne facilitera pas forcément le portage des applications existantes sous Turbo Vision : il conviendra de repenser complètement les programmes mêlant intimement la gestion des interfaces «utilisateur» et le traitement applicatif même! Si le Turbo Pascal orienté objet ne génère pas de code pour Windows (Turbo Pascal for Windows existe par ailleurs) et autorise encore moins la création de bibliothèques dynamiques (les fameuses DLL), la présence de Turbo Vision permet d'assurer, en mode texte, le développement d'applications homogènes.

C++ ou Turbo Pascal orientés objet?

Avec le Turbo Pascal orienté objet, les inconditionnels du pascal n'auront plus à souffrir de l'abandon d'un langage aussi ergonomique pour goûter aux joies de la programmation par objets (grâce en partie à Turbo Vision). Certes, les zélateurs du langage C++ critiqueront le côté non automatique du mécanisme de constructeurs-destructeurs en TPOO, l'impossibilité de surcharger les opérateurs ou encore (de façon plus méprisante) l'absence d'héritage multiple. C'est un fait : le TPOO n'a pas la puissance du C++ (ni sa complexité) et il ne concerne que les développeurs en environnement DOS.

Néanmoins, associé à Turbo Vision, il constitue une bonne introduction au style «objet» et à la programmation par événements avant d'attaquer, par exemple, un monstre tel que Windows en programmation.

Le Borland C++ s'adresse, quant à lui, plus particulièrement aux programmeurs «Windows» avertis mais quelque peu échaudés par la complexité du SDK de Microsoft. En résumé, le C++, langage de professionnels de l'industrie, trouve en Borland 2.0 un outil de développement à sa mesure pour des applications Windows-DOS.

Le pascal, langage universitaire par excellence, fait son entrée dans le monde des objets grâce à TPOO et à Turbo Vision : outils assurant, dès maintenant, l'assimilation des techniques de programmation des années 90.

C-scape : un véritable plaisir.

Gestionnaire d'interfaces écran graphique et texte

Le système de gestion d'écrans C-scape® est une bibliothèque de fonctions en langage C permettant la saisie et la validation de données à l'écran, la gestion de menus et fenêtres, l'édition de texte, et les systèmes d'aide en ligne. L'outil de conception d'écrans de C-Scape, Look & Feel™, vous permet de créer des écrans de façon interactive et de générer le code source correspondant.

C-Scape comprend des fonctions de haut niveau facilement modifiables, aussi bien que des primitives permettant de construire de nouvelles fonctions. Grâce à l'architecture orientée objet de C-Scape, vous développerez des applications plus fonctionnelles, plus flexibles, plus portables, et vous aurez plus de plaisir à le faire.

Un produit phare dans l'industrie du logiciel. Depuis 5 ans, plusieurs milliers de développeurs dans le monde entier se sont convertis à C-Scape, abandonnant leur bibliothèque "maison", ou des produits peu flexibles. La presse approuve: "C-Scape est de loin le meilleur. ...un véritable plaisir", ecrit IEEE Computer. Les plus grandes sociétés ont selectionné C-Scape comme standard de développement.

C-Scape peut être utilisé avec des SGBD, des bibliothèques graphiques, ou toute bibliothèque C ou C++. De plus, C-Scape peut combiner texte et graphique. Pour porter votre application de DOS ou OS/2 vers UNIX, VMS, AIX, recompilez simplement. C-Scape supporte également les "DOS extenders" Phar Lap, Rational et Ergo, ainsi que MetaWINDOW.

Graphique. Fonctionne en couleur, en mode texte ou graphique. Peut lire des images graphiques depuis des fichiers PCX.

Architecture orientée objet. Ajoutez vos propres fonctions et créez des modules de code réutilisables.

Support de la souris complètement intégré pour la selection des options de menu, les champs de saisie, et pour déplacer et redimensionner une fenêtre.

Portabilité. Le code de C-Scape est indépendant du hardware. Supporte DOS, OS/2, UNIX, AIX, VMS, QNX, XENIX... Détecte automatiquement Hercules, CGA, EGA, VGA, et les modes 43 et 50 lignes.

Edition de texte. Comprend un éditeur de texte avec contrôle de la césure, fonctions rechercher/remplacer, et manipulation des blocs de texte.

Flexibilité des champs. Champs masqués, protegés, marqués, avec label, obligatoires, avec ou sans echo. Validation complète des données. Champs de type heure, date, montant, liste déroulante, et autres fonctions de haut niveau. Créez votre type de champ.

Fenêtres. Encadrées, pop-up, superposées, eclatées, ascenceur, défilement horizontal; taille et nombre des fenêtres limités par la memoire RAM uniquement.

Menus. Déroulants, hiérarchiques, pop-up, style Lotus 1-2-3, ou créez le vôtre.

Système d'aide contextuel. Prompts et messages d'aide attachés à un écran ou à un champ. Messages à références croisées pour créer un système d'aide en ligne de style hypertexte.

Gestion des écrans. Ecrans stockés dans des fichiers ou linkés avec l'application.

Génération de code. Créez n'importe quel écran avec l'outil interactif Look & Feel, testez-le, puis convertissez-le automatiquement en code C.

Essayez C-Scape ! C-Scape est *puissant, flexible, portable*, d'apprentissage et d'utilisation facile. C-Scape est livré en standard avec la totalité du code source de la bibliothèque et l'outil de création d'écrans Look & Feel. La documentation est claire, extensive et detaillée. Avec C-scape, pas de module résident et pas de royalties sur les applications diffusées.

C-Scape est un produit d'**Oakland Group**. Smartline assure le support technique de C-scape par téléphone et par fax ainsi que des sessions de formation sur le produit.

Appelez-nous au (1) 46 34 64 19
ou envoyez le bon de commande ci-dessous par fax ou par courrier. Après le plaisir de C-Scape, le développement ne sera jamais plus ce qu'il était.

C-scape et Look & Feel **version 3.2** : DOS, OS/2 (Borland, Microsoft, Zortech): 4500FHT; UNIX: 13500FHT; Autres OS: nous consulter. Cartes de crédit acceptées.

smartline

30, rue Saint André des Arts
75006 Paris France
Fax: (1) 40 46 85 89 ***Tél: (1) 46 34 64 19***

Oakland Group, Inc 675 Massachusetts Ave, Cambridge, MA 02139 USA.Tél: 617-491-7311. Les produits d'Oakland sont distribués en Australie, Belgique, Danemark, Autriche, Allemagne, Suisse, Norvège, Suède, Angleterre, France. La photographie représente un programme C-Scape combinant une saisie de données avec des images vidéo chargées depuis un fichier .PCX. C-Scape et Look & Feel sont des marques déposées par Oakland Group, Inc.; les autres marques citées appartiennent à leurs priopriétaires respectifs. Les fonctionnalités, prix et conditions sont sujets à modification. Documentation en anglais. C-scape (seascape): paysage de mer.

Service lecteurs n° 13 (dernière page)

✂

❑ Je désire recevoir une documentation sur C-scape

❑ Je désire recevoir le dossier d'évaluation comprenant les deux disquettes de démonstration C-scape et Look & Feel. Ci-joint un chèque de 120 F TTC. Une facture sera jointe à l'envoi.

nom prénom _____ société _____ tél _____

adresse _____

LANGAGE

LES PROFILERS
Microsoft contre Borland

ALEXANDRE COSTE

Qu'est-ce qu'un analyseur de performances (également connu comme moniteur de performances), plus communément appelé «Profiler»? Il s'agit d'un outil logiciel qui mesure l'efficacité d'un programme en détectant ses points faibles *(Cf.encadré «Ce que contrôle un optimiseur»)*. Il contrôle les activités essentielles et fournit des rapports statistiques détaillés touchant à l'efficacité du programme. Il guide l'utilisateur à l'intérieur des programmes pour en étudier les opérations les plus subtiles, à savoir les temps d'exécution, le comptage des instructions, les appels d'interruption et les accès aux fichiers.

De multiples plates-formes pour le Microsoft Source Profiler

Le Microsoft Source Profiler (MSP) présente la particularité de pouvoir travailler sous plusieurs systèmes d'exploitation, de la version 3.30 de MS-DOS à OS/2 (1.2x et 1.3) en passant par Windows 3.0. La version de Turbo Profiler (TP) de Borland que nous avons testée ne fonctionne, pour sa part, que sous MS-DOS 2.0 ou ultérieur (Borland annonce la disponibilité d'une version 2.0 pouvant s'exécuter sous Windows). Le MSP requiert 384 ko de mémoire vive pour s'exécuter sous MS-DOS, 4 Mo pour OS/2 et 1 Mo pour Windows. Il occupe, en revanche, 2 Mo sur le disque dur contre 600 ko pour le TP. Le MSP ne fonctionne que sur les machines équipées d'un 80286 ou d'un processeur plus puissant (un 80386 est fortement conseillé). La documentation précise que de mauvais résultats peuvent être obtenus si l'ordinateur possède un cache-mémoire de taille importante. Quant au TP, il peut s'exécuter quel que soit le microprocesseur 80x86.

CE QUE CONTRÔLE UN OPTIMISEUR

Un Profiler contrôle les performances du programme :
- les points où le programme consomme beaucoup de temps,
- le nombre de fois qu'une ligne est exécutée,
- le nombre d'appels faits à un sous-programme et l'origine de l'appel,
- le nom des fichiers le plus souvent utilisés et la durée d'ouverture.

Il contrôle également les ressources matérielles, à savoir :
- le temps de traitement du processus central,
- les accès disques,
- les saisies clavier,
- les sorties imprimante,
- la nature des interruptions.

LES LANGAGES SUPPORTÉS PAR LES OPTIMISEURS

Le Microsoft Source Profiler 1.0 peut s'exécuter sous MS-DOS, OS/2 1.2x et Windows. Il reconnaît les langages suivants : Basic 6.0 et plus, QuickBasic 4.0 et plus, C 5.1 et plus, Quick C 2.0, Quick C avec Quick Assembleur version 2.01 et plus, Cobol 4.0 et plus, Fortran version 5.1 et plus, Masm version 5.1 et plus, Pascal version 4.0 et plus. Le Turbo Profiler 1.0 s'exécute uniquement sous MS-DOS et reconnaît : Turbo Pascal version 4.0 et plus, Turbo C version 2.0 et plus, Turbo Assembleur 1.0 et plus.

Les restrictions

Le MSP n'optimise pas des programmes sous MS-DOS qui ont recours aux overlays (unités de recouvrement). Il en va de même pour le programme dont l'exécution s'opère sur 16 ou 32 bits, lorsque le circuit de l'horloge (8253) est reprogrammé ou bien lorsque les interruptions INT 01H et 03H sont utilisées et, enfin, quand le programme est résident. D'après Borland, le TP travaille dans ces conditions. Le MSP rencontre également des problèmes sous OS/2 : il ne fonctionne pas sous les versions 1.1x. Son utilisation n'est possible qu' avec les versions 1.2x et 1.3, sous réserve d'ajouter dans le fichier Config.Sys la déclaration IOPL = YES. Quant à

A mesure que les applications gagnent en puissance et en richesse fonctionnelle, le code exécutable correspondant prend de l'embonpoint. Il n'est pas rare aujourd'hui que des applications DOS dépassent les 600 ko voire le mégaoctet sous Windows. Même si la mémoire vive n'est plus une denrée aussi précieuse et coûteuse qu'il y a quelques années, il reste intéressant de pouvoir optimiser le code des programmes afin d'en augmenter la vitesse de traitement ou bien de réduire la taille du code.

Windows 3.0, il impose lui aussi des limitations, identiques à celles existant sous MS-DOS. La documentation précise encore que de mauvais résultats peuvent être obtenus : Windows étant multitâche, la mémorisation des temps d'exécution peut être faussée. Le MSP ne fonctionne pas en mode réel, et toutes les librairies DLL doivent avoir leur propre routine d'initialisation. Enfin, si le programme fait appel à la fonction GlobalWireWin, l'optimisation reste impossible... Au vu de toutes ces restrictions, qu'apporte donc le MSP? Il supporte quantité de langages dans les trois systèmes d'exploitation *(Cf.encadré «Les langages supportés par les optimiseurs»)*.

Le MSP et le TP analysent tout programme qui a été compilé avec les options permettant de le tracer sous CodeView. Borland fournit dans ce sens deux utilitaires qui modifient les exécutables créés en Microsoft C ou Microsoft Assembleur afin de les rendre accessibles au Turbo Profiler (ces deux produits sont TDMAP et TDCONVERT fournis avec le logiciel Turbo Debugger de Borland International).

Différentes méthodes d'optimisation

Le MSP propose quatre méthodes d'optimisation :

❑ **Counting** : cette méthode détermine combien de fois une même partie de code est exécutée. Elle peut s'avérer utile pour la détection d'une fonction récursive.

❑ **Coverage** : cette méthode est la plus rapide mais la moins performante. Elle enregistre quelles lignes ont été exécutées pendant une session de travail mais ne garde pas les informations utiles (tailles et temps d'exécution). Elle permet de repérer dans une masse de codes importante quelles sont les zones de Dead Code, c'est-à-dire celles qui ne s'exécutent jamais.

❑ **Timing** : quand l'optimisation est en cours, les informations sur la durée d'exécution sont mémorisées. Cette méthode permet de savoir combien de fois une séquence de codes s'est exécutée.
Elle est la plus lente car le timer doit relever le temps chaque fois qu'une ligne, une fonction ou une procédure sont appelées. Pour une efficacité accrue, il est conseillé d'utiliser cette méthode uniquement sur une région précise de programme.

❑ **Sampling** : c'est un type d'optimisation simple. Quand cette option est sélectionnée, le MSP arrête l'exécution du programme en cours et mémorise à des intervalles de temps réguliers quelle est la procédure, la fonction ou l'instruction qui se trouve exécutée. Les

EXEMPLE DE DOCUMENT ÉDITÉ PAR MICROSOFT

Exemple de document édité par Microsoft Source Profiler après une analyse du programme à optimiser.
Statistiques globales du programme :
- le temps total d'exécution,
- le temps pendant lequel le programme s'est exécuté jusqu'à la première zone sélectionnée pour l'optimisation,
- le nombre d'appels effectués par le programme,
- le nombre total de fonctions présentes dans le programme,
- le nombre total de fonctions marquées pour l'optimisation,
- le pourcentage d'évolution sur le nombre de fonctions à optimiser sur deux sessions de travail.

Statistiques sur le programme exécutable :
- le temps de chargement de l'exécutable (ou COM ou DLL),
- le pourcentage d'occupation de ce module par rapport à d'autres éventuels,
- le nombre de fonctions présentes dans ce module,
- le nombre de fonctions de ce module marquées pour l'optimisation,
- le pourcentage d'évolution sur le nombre de fonctions de ce module à optimiser sur deux sessions de travail.

Table des fonctions :
- le temps d'exécution de la fonction,
- le pourcentage d'occupation par rapport au code total,
- le temps pris par la fonction pour en appeler une autre,
- le nombre de fois que la fonction est appelée,
- le nom de la fonction.

Programme de calcul et affichage des 1 000 nombres premiers non optimisé

PREMIER1.C

```
#include <sstdio.h>
void premier(int n)
        {
                int i;
                for (i=2; i< n; i++)
                        if (n % 1 == 0)
                                return 0;
                        return 1;
        }
void main(void)
        {
                int i, n = 2000;
                for (i=2; i<=n; i++)
                        if (premier(i))
                                printf("%d\n",i);
        }
```

Même programme mais optimisé

PREMIER2.C

```
#include <stdio.h>
#include <conio.h>
void premier(int n)
        {
                int i;
                if (n % 2 == 0) return (n==2);
                if (n % 3 == 0) return (n==3);
                if (n % 5 == 0) return (n==5);
                for (i=7;i*i <= n; i+=2)
                        if (n % i == 0)
                                return 0;
                        return 1;
        }
void main(void)
        {
                int i,n = 2000;
                for (i=2;i<=n;i++)
                        if premier(i))
                                cprintf("%d ", i);
        }
```

résultats obtenus peuvent donner une idée approximative du temps d'exécution global. Borland propose des méthodes voisines :

❑ **Time** : affiche la durée totale passée dans chaque zone marquée (option sélectionnée par défaut).

❑ **Counts** : affiche le nombre total de fois où le déroulement du programme s'est exécuté dans chaque zone.

❑ **Both** : regroupe les temps et les passages sur un même écran.

❑ **PerCall** : affiche la durée moyenne par appel de routines.

❑ **Longuest** : affiche la durée maximale passée dans chaque zone sélectionnée. La méthode préconisée par les deux optimiseurs consiste à sélectionner le comptage et le temps mis comme critères d'analyse. Grâce aux informations recueillies, le développeur peut vérifier si le programme ne fait «pas plus de travail» que prévu.

MSP exporte vers Excel

Le MSP édite un rapport selon plusieurs critères *(Cf.encadré «Exemple de document édité par Microsoft»)*. Le document obtenu peut être sorti sous la forme d'un fichier ASCII ou sur imprimante. Dès lors, pour permettre au développeur de se faire une idée plus pragmatique de l'état des fonctions, le Microsoft Source Profiler propose une idée originale qui consiste à transformer le fichier ASCII en un fichier lisible par le tableur Excel 3.0. Cette action est possible grâce à l'option /T du programme PLIST fourni avec le MSP (exemple : PLIST /T QSORT.PBT >QSORT.TAB).

Une fois récupéré sous Excel, ce fichier est mis en forme à l'aide de la macro PROFILER.XLM qui produit un superbe graphe illustrant l'état actuel des fonctions selon le critère retenu. Le TP n'offre pas une telle facilité, mais il apporte bien d'autres outils qui sont loin d'être négligeables pour le confort du programmeur.

Citons l'option de «téléanalyse» qui consiste à relier deux ordinateurs : l'un fait tourner le Turbo Profiler tandis que le programme à optimiser s'exécute sur l'autre. TP propose aussi l'analyse en mode virtuel 80386 qui passe la barre des 640 ko de mémoire. Cette option est utile pour l'optimisation de programmes de grande taille et permet de détecter des bogues qui ne pourraient être détectés si le programme se trouvait chargé en mémoire haute.

Elle autorise également le suivi des opérations de lecture et d'écriture en mémoire ou sur les ports de sortie tout en conservant (presque) la vitesse maximale de traitement. Elle est, en revanche, inopérante si DesqView, Windows/386, QEMM 386 ou 386MAX sont présents en mémoire.

Enfin, il est possible de travailler avec deux sources d'affichage différentes (par exemple, une carte VGA et une carte Hercules) : l'optimisation du programme s'opère alors sur l'écran monochrome tandis que l'application s'exécute sur l'écran VGA, ou vice versa.

Les programmes ci-contre donnent un exemple de l'optimisation obtenue sur un programme qui cherche et affiche les mille premiers nombres premiers (Quick Sort). Le programme C non optimisé représente la synthèse des exemples de démonstration de Microsoft Source Profiler et de Turbo Profiler.

Le listing est loin d'être optimisé. Le résultat final obtenu grâce aux informations délivrées par un optimiseur est donné dans le programme optimisé. Il s'exécute en 0,5264 seconde contre 4,5675 secondes avant optimisation.

DATABOSS: LE GENERATEUR COMPLET DE VOS APPLICATIONS EN PASCAL OU C

Avec DATABOSS, gagnez des semaines de développement.

DATABOSS est un environnement complet conçu pour le développement de bases de données relationnelles pour des applications en Turbo Pascal ou en C.
DATABOSS permet de générer les différentes parties de votre application: menus, fichiers (avec linkage), écrans d'entrées de données, état, réindexation et modules de reconfigurations, rapports.

DATABOSS produit du code source Pascal ou C pour des applications mono ou multi-utilisateurs. Les écrans peuvent être multi-fenêtres, comporter des champs mémos, ainsi qu'une aide en ligne et des champs virtuels ou calculés.

Son nouveau générateur d'état WYSYWYG vous permet de créer vos rapports très facilement.

De plus DATABOSS ne nécessite aucune connaissance de développement.
Sa documentation en français comporte plus de 300 pages et le support technique est gratuit.

La version de démonstration est limitée à 10 utilisations et l'ensemble des manuels est disponible à **490 F HT.**

Version 3.0 en C. et 3.5 en Pascal

Prix intro: ~~7 995~~ F HT
5 995 F HT TVA 18,6%
jusqu'au 31/03/92

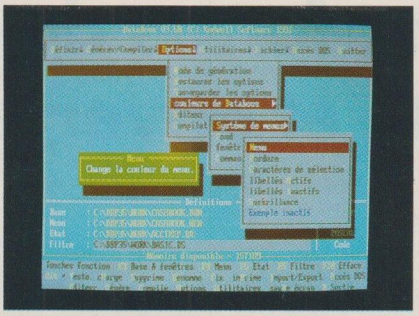
Pour vous renseigner ou commander
Téléphonez au: (1) 40 99 28 00
Fax: (1) 40 99 28 88

INNOSOFT

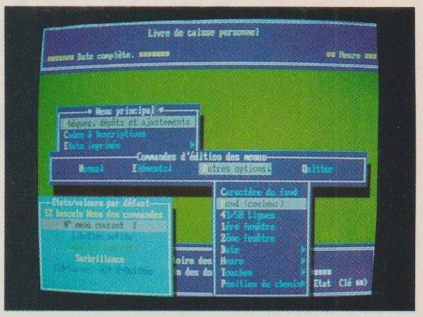
RENDEZ-VOUS PC FORUM 11 - 14 Février 1992 PARIS Porte de Versailles HALL 7 STAND

STAND W47

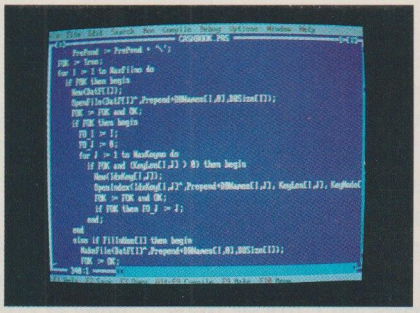
2, rue des Bourets - 92150 Suresnes



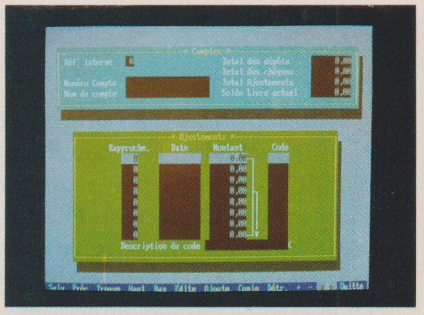
Un générateur complet.



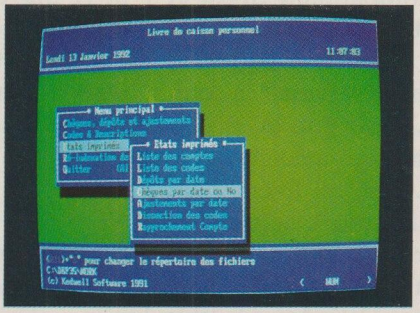
Une interface conviviale qui facilite votre travail.



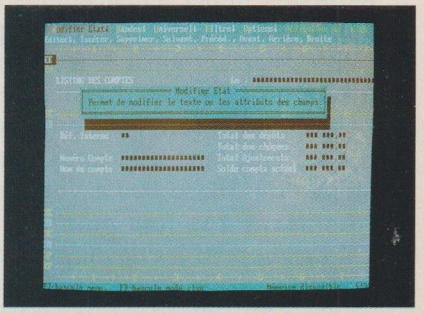
Génération de code source compilable - sans royalties.



Une gestion de base de données ISAM.



Menus déroulants, support souris, écrans de saisie, multifenêtres, champs mémo et déroulables.

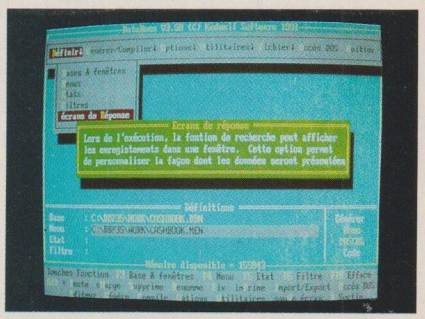


Un générateur d'état automatique en WYSYWYG (version 3.5).

DATABOSS

- *Génère du code C ou Pascal*
- *Compatible TP55 et 6, Quick Pascal, Turbo C, C++*
- *Compatible Novell, 3 COM, Netbios*
- *Générateur de rapport WYSYWYG (3.5)*
- *Pas de royalties*
- *Pas de nécessité de savoir programmer*
- *Aide en ligne*
- *Importation directe possible depuis dBASE.*

Service lecteur N° 75



Une aide contextuelle à tout niveau simplifie le développement.

Service lecteurs n° 14 (dernière page)

BON DE COMMANDE/DEMANDE DE DOCUMENTATION
A renvoyer à INNOSOFT - 2, rue des Bourets - 92150 Suresnes

☐ je commande **la version d'évaluation de DATABOSS** avec ses manuels au prix de **652,30 F TTC** (soit 581,14 F TTC + 71,16 F TTC de frais de port).
☐ **OUI** je désire recevoir une documentation commerciale détaillée.

Ci-joint mon règlement:
☐ Chèque ☐ Carte Bleue N° ..... Date d'expiration.....
☐ Contre-remboursement (60 F TTC en sus).

Nom: ..... Adresse: .....
Société: ..... Code postal: .....
Ville: ..... Tél.: .....

PCE 3/92

LANGAGE

CLIPPER 5.0

Une plate-forme de développement puissante et ouverte

GILBERT CHAUVAUX

Clipper utilisait à l'origine le langage dBase III plus, à l'exclusion des commandes interactives (comme EDIT), et travaillait sur des fichiers au format dBase. Clipper apporta des améliorations au langage, comme les menus, les tableaux, le support des réseaux locaux et, surtout, les «fonctions définies par l'utilisateur» ou UDF (User-Defined Functions), écrites en Clipper mais aussi en C et en Assembleur. Rapidement, les utilisateurs de Clipper renoncèrent à la compatibilité dBase et aux avantages du langage interprété.

Ils donnaient ainsi le feu vert à Nantucket pour une évolution ultérieure du langage qui préserverait la compatibilité des formats de fichiers, ainsi que le «style» dBase, cette facilité et cette puissance qui avaient fait le succès de dBase II. La version 5.0 est un événement important à la fois pour les utilisateurs actuels de Clipper et pour les développeurs micros en général.

Elle constitue actuellement la plate-forme de développement d'applications de gestion hors-Windows la plus puissante et la plus ouverte qui soit. Encore faut-il bien comprendre et savoir utiliser les apports de cette version — et avoir conscience de l'effort à y consacrer. Nous allons rapidement présenter les nouveautés de Clipper 5.0, quitte à les reprendre en détail dans des articles futurs, et tenter d'évaluer ce nouveau langage.

La mémoire virtuelle

Comme tous les langages récents destinés au monde DOS, la version 5.0 résout le problème de la limitation mémoire qui obligeait le programmeur à structurer les grosses applications en «overlays». La version 5.0 utilise la technique de la mémoire «virtuelle». Le programme et les données qu'il manipule en mémoire sont quasi illimités, étant «réellement» implantés en mémoire vive, mémoire EMS (expanded), et sur disque. Code et données sont automatiquement gérés par pages de 1 ko.

Les données sont «swappées» au besoin sur disque, et le code est relu, à la demande, à partir du fichier .EXE . La mémoire virtuelle est réalisée dans la version 5.0 par :

- le gestionnaire de mémoire virtuelle (VMM : Virtual Memory Manager);
- l'éditeur de liens livré avec Clipper : RTLINK, utilisant la technique d'«overlay dynamique», ou de tout autre éditeur de liens compatible. Il permet également le link «incrémentiel» réduisant le temps d'édition de liens et l'utilisation de librairies «prélinkées».

Il s'agit d'un fichier .PLL contenant une grande partie de ce que tous les fichiers exécutables .EXE ont en commun (plus de 150 ko de code).

Ce code est «retiré» des fichiers .EXE et lu, lors du chargement, dans le fichier .PLL. Le gain de place en occupation disque est important si l'on a plusieurs .EXE Clipper.

Support de bases de données non-dBase

Un apport non négligeable de la version est la possibilité — non encore délivrée par Nantucket — de «moteurs» de bases de données «remplaçables».

En clair, cela signifie qu'une application Clipper pourrait attaquer simultanément plusieurs bases de données, dBase ET autres : Oracle, SQL Server, etc., et cela, en utilisant les ordres issus de dBase que connaît le programmeur, donc de façon transparente.

Nous verrons que la possibilité de redéfinition des commandes déja incluse dans la version 5.0 autorise déja de telles pratiques.

La mise en œuvre des RDD (Replaceable Device Drivers) passe en fait par la mise à disposition par Nantucket d'une API (Application Programming Interface) qui permettra à des sociétés tierces de réaliser des moteurs qui seront commercialisés.

La version 5.0 facilite l'écriture et la mise au point des programmes en fournissant une aide en ligne sur le langage (basée sur les Norton Guides), ainsi qu'un débogueur au niveau source de la qualité des modèles modernes, comme celui de Microsoft (CodeView).

La version 5 apporte un paramétrage supplémentaire

Conçu à l'origine comme compilateur dBase, Clipper de Nantucket est devenu, dans sa dernière version, un langage à part entière. Il apporte des extensions puissantes telles que les commandes définies par l'utilisateur, la programmation objet ou les blocs de code. Il permettra bientôt de traiter avec les mêmes commandes les bases de données non-dBase.

de la compilation — le compilateur étant, par ailleurs, optimisé — avec des possibilités de contrôle de déclaration de variables. A l'exécution, l'utilisation de la mémoire, conventionnelle et étendue, est également paramétrable.

Les modifications du langage

Jusqu'ici, il n'a été question que d'apports qui ne remettent pas en cause la façon de programmer en Clipper. La version 5.0 possède des nouveautés allant du détail à la programmation objet.

Bien entendu , tous les éléments du langage Clipper de la version précédente (summer 87) sont intégralement repris en 5.0, de sorte que le programmeur peut ignorer les nouveautés.

La version 5.0 intègre cette facilité bien connue des programmeurs C, avec une extension importante : la possibilité de redéfinition des commandes. Le préprocesseur permet la définition de constantes manifestes, de macros, d'inclusions de fichiers et de compilations conditionnelles. Une commande définie par l'utilisateur (UDC) est une véritable définition de syntaxe. Par exemple :

```
#command SELECT <champ,...>
FROM <table> [WHERE <condition>] =>;
USE <table> NEW;
LIST <champ> [FOR <condition>];
USE
```

L'exemple ci-dessus définit une nouvelle commande de type SQL : SELECT. La première ligne définit la syntaxe d'écriture de cette commande, avec les parties variables (identifiées par <...>) et optionnelles([...]), et sur la seconde ligne, après «=>», ce que le préprocesseur doit générer. Comme en pascal et en C, les commandes sont séparées par des points-virgules «;».

Une nouveauté fondamentale — et surprenante — vient de ce que le langage Clipper 5.0 est lui-même défini à l'aide de ces directives #command! De plus, le fichier de définition est donné et peut être modifié.

Le langage «ultime» 5.0 se réduit aux instructions de tests, boucles... et à l'appel de fonctions de la librairie Clipper. Certaines sont «publiques», c'est-à-dire supportées et documentées, et d'autres «privées», utilisables sans documentation ni certitude de support dans les versions ultérieures.

Les applications sont immenses : amélioration du langage, personnalisation, traduction, compilation d'autres langages (un kit de compilation de dBase IV est déja disponible) et modification du code généré.

Les commandes dBase de fichiers peuvent, en particulier, se traduire par l'utilisation d'autres types de bases de données par l'intermédiaire de librairies de fonctions d'accès, comme dans l'exemple (fictif) :

```
#command LIST <alias> -> <champ>
[FOR <condition>] =>;
SQLCall("SELECT <champ>
FROM <alias> [WHERE <condition>]")
```

Le look C

Les réalisateurs de Clipper sont des programmeurs C (le compilateur et les librairies sont écrits en C), et y ont été apporté, en plus du préprocesseur, d'autres éléments issus de C, notamment les variables à portée «lexicale». Celles-ci doivent être déclarées en début de procédure, fonction ou source.

Contrairement aux variables traditionnelles de dBase, publiques ou privées, elles ne sont visibles que dans la fonction ou le source (variables dites «statiques externes»). Parmi ces variables, les «statiques» gardent leurs valeurs quand on revient dans la fonction ou le source. Ces variables «locales» et «statiques» de la version 5.0 sont gérées plus efficacement que les autres. Elles ne peuvent provoquer de «collisions» indésirées comme c'est le cas des variables usuelles. Elles sont déclarées par :

```
LOCAL  x,y,z:=1
STATIC u:="abc"
```

Les autres apports sont :

❑ La déclaration des paramètres de fonctions «en ligne» :

```
function  f1(x,y,z)
```

Ceci remplace l'instruction PARAMETERS, les paramètres étant alloués comme variables locales.

❑ Les fonctions «statiques» visibles uniquement dans leur source et ne pouvant donc pas interférer avec d'autres fonctions :

```
STATIC function  f2(x,y,z)
```

❑ L'écriture de plusieurs instructions sur une ligne :

```
a=0 ; b=0 ; c=0
```

❑ Les commentaires délimités par // et /* .. */

❑ Les opérateurs d'affectation en ligne, «compound», pré- et post-incrémentation-décrémentation :

```
a += b := c := ++d
```

La ligne ci-dessus incrémente d, affecte le résultat à c et b et l'ajoute à a.

❑ L'opérateur de «concaténation» d'expression «,»: (<exp1>,<exp2>,<exp3>) évalue les expressions <exp1> et <exp2> (qui entraîneront des effets de bord) et vaut <exp3>.

❑ Une nouvelle conception des tableaux Clipper qui réalisent à la fois les tableaux multi dimensionnels et les structures C. De plus, les tableaux sont systématiquement traités par pointeurs dans la version 5.0 (ce qui n'est pas le cas de C), ce qui ouvre des possibilités de pointeurs en Clipper.

Les tableaux de la version 5.0

Les tableaux 5.0 sont multidimensionnels uniquement par le fait qu'un élément de tableau peut lui-même être un tableau, dont chaque élément à son tour... Voilà plusieurs déclarations et utilisations de tableaux :

```
local t1[3,2],t2:={{11,12},{21,22}},t3,t4
t1[1,1] = 11; t1[1,2] = 12;
t1[2,1] = 21; t1[2,2] = 22;
t3 = {1,{2,"trois"},
t2[2]}
t4 = t3
t4 = aclone(t3)
```

Dans Clipper 5.0, les déclarations peuvent être accompagnées d'affectations par l'opérateur «:=». Les expressions entre accolades {..,..,..} sont des tableaux «littéraux». Dans l'exemple ci-dessus t1 et t2 ont exactement le même contenu de quatre éléments. t3 est un tableau «à géométrie variable» propre à Clipper et s'apparente aux structures C ou pascal : il est constitué d'un premier élément scalaire, d'un second élément qui est un tableau à deux éléments de types différents et d'un «pointeur» vers le second élément de t2, qui n'est qu'un autre tableau à deux éléments! En 5.0, les références à des tableaux sont systématiquement considérées comme des pointeurs.

Ainsi, t4=t3 ne fait qu'affecter à t4 un pointeur, et non une copie de tableau. t3 et t4 représentent alors le même tableau. Si l'on veut vraiment copier un tableau, on utilise la fonction ACLONE(). Les fonctions Clipper 5.0 peuvent renvoyer des tableaux.

La version 5 favorise la déclaration des variables avec les variables LOCAL et STATIC. De plus, les autres variables et même les champs peuvent être déclarés avec de nouvelles déclarations MEMVAR et FIELD. Un paramètre de compilation permet d'être prévenu quand un symbole non déclaré apparaît, évitant ainsi des erreurs lors de l'exécution.

Selon la philosophie dBase, la déclaration ne fait que signaler l'existence d'un symbole sans préciser, comme en C, le type d'information contenu, qui reste dynamique.

La programmation objet en Clipper 5.0

Il y a peu de produits de développement récents qui ne soient teintés d'orientation objet. La version 5.0 n'y fait pas défaut, en offrant une possibilité limitée de POO. La limite est liée à ce qu'on dispose de quatre classes prédéfinies, non modifiables et sans possibilité d'en rajouter.

Ces classes sont :

❑ La classe ERROR définissant des objets «erreurs», ne faisant que transporter l'information liée aux erreurs à l'exécution;

❑ Les classes TBROWSE et TBCOLUMN contrôlant l'affichage de tableaux, le «browsing» de fichiers, etc., avec des possibilités étendues : contrôle total des actions clavier, restructuration dynamique des colonnes, etc.

Voici un exemple d'utilisation de la classe TBROWSE, faisant apparaître en particulier l'opérateur d'envoi de messages <objet>:<message>(arguments,..) :

```
tb = TBrowseNew(10,10,20 60)
// coordonnées d'affichage
// tb est un OBJET de la classe TBrowse
tb:freeze  = 1
// on  met 1 dans la "variable 'instance"
// freeze:
// nombre de colonnes bloquées
cl = TBColumnNew(...)
// objet colonne
tb:insColumn(3,cl)
// insertion d'une colonne en utilisant la
```

```
"méthode" insColumn()
```

Des extensions commercialisées aux Etats-Unis comme Class(y) ou développées en France par l'auteur de cet article permettent de s'affranchir des limites Clipper en matière de POO en autorisant la création de classes nouvelles avec héritage. De plus la prochaine release de Clipper 5 sera dotée de ces extensions.

Les blocs de code

Le bloc de code est la nouveauté la plus exotique de la version 5.0. C'est un élément qui surprendra les programmeurs, excepté ceux qui connaissent Smalltalk, langage de référence de la POO, dans lequel existe une notion très semblable : le bloc de «contexte». Un bloc de code est l'équivalent limité d'une fonction recevant des paramètres et retournant une valeur :

```
b = {|x,y| upper(x)<upper(y)}
```

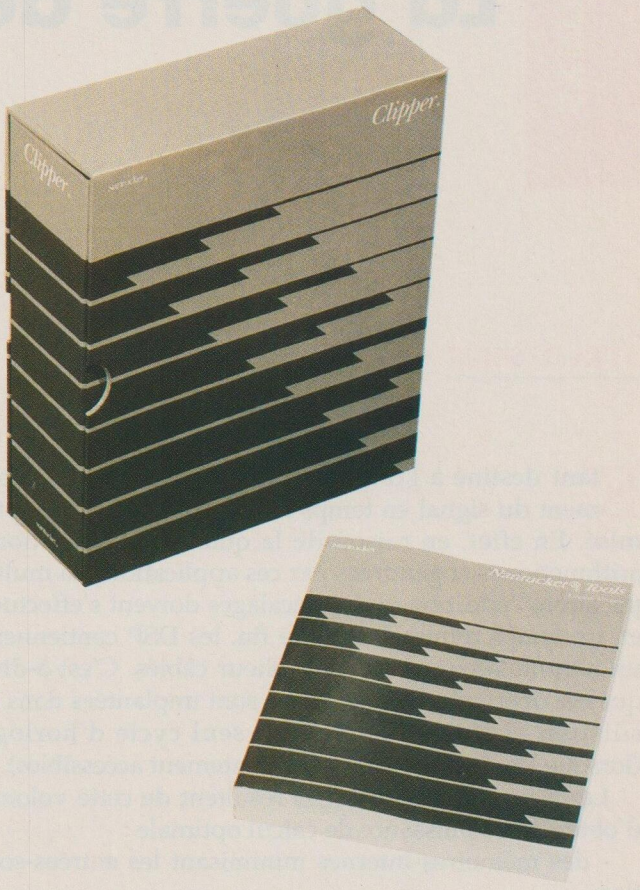
Ce bloc reçoit deux paramètres x et y et retourne le résultat de l'opération de comparaison : la valeur logique vrai ou faux selon que x (supposé être une chaîne de caractères) est avant ou après y dans l'ordre alphabétique sans tenir compte des majuscules. Le bloc de code est précompilé et manipulé comme une autre donnée en étant affecté à une variable puis passé à une fonction .

Il est exécuté par : eval(b, "aa","Bc") qui renvoie ici la valeur vraie. L'intérêt du bloc de notre exemple apparaît dans un programme de tri de tableaux qui est paramétré pour effectuer des comparaisons d'éléments, cette façon lui étant indiquée par un bloc de code.

C'est en fait ce que réalise la fonction de tri fournie avec la version 5.0. Les blocs de code sont très utilisés de façon interne par Clipper et dans les classes GET et TBROWSE. Ils offrent une possibilité accrue de paramétrage, peuvent contenir des contrôles et des actions à effectuer en association avec des événements : touches, souris, etc. Ils se substituent efficacement aux macros issues de dBase.

Il y a bien d'autres apports de la version 5.0 qui permettent de savoir si une variable a été initialisée ou s'il manque un paramètre dans un appel de fonction, quantité de nouvelles fonctions, quelques modifications de commandes, des fonctions réseau, un système de gestion d'erreurs remanié entraînant un contrôle total à l'exécution, une clause WHEN vérifiant l'entrée dans une zone de saisie, etc.

Même si l'on ne change pas sa façon de programmer, Clipper 5.0 apporte des améliorations notables à la version précédente : la mémoire virtuelle, le débogueur source et la promesse du support d'autres bases de



Clipper. Une version aux apports considérables.

données. Les nouveautés du langage apportent une puissance considérable au développeur, en matière de productivité, de généricité et de réutilisation des sources. Cette puissance coûte cher en temps d'apprentissage, d'élaboration de fonctions de base, etc.

On aboutit ainsi à un langage à plusieurs «vitesses»: une utilisation de type dBase, pour une programmation «vite fait» traditionnelle n'employant pas les nouveautés, une utilisation «système» pour faire des outils généraux, beaucoup plus complexes, réclamant une grande fiabilité, des déclarations et, enfin, une utilisation basée sur des objets, des fonctions et des commandes résultant de la programmation système, dans laquelle le programmeur ne sait pas forcément tout ce qui se passe «en dessous» et peut même ignorer dBase.

Avec la version 5.0, Nantucket annonce franchement qu'il se démarque du monde dBase en s'orientant vers une architecture ouverte, un langage déclaratif plus rigoureux et la POO. Clipper restera lié au DOS en mode texte.

Une autre ligne de produits futurs connue sous le nom de Aspen reprendra les mêmes principes, mais en s'émancipant de la compatibilité dBase et même de Clipper au niveau langage, et fonctionnera, entre autres, sous Windows.

SYSTÈME

DSP CONTRE RISC
La guerre des Mflops

ATTA OLOUMI

Etant destiné à l'origine aux applications de traitement du signal en temps réel, le DSP doit être optimisé. En effet, en raison de la quantité d'opérations arithmétiques engendrées par ces applications, la multiplication, l'addition ou les décalages doivent s'effectuer en un temps minimal. A cette fin, les DSP contiennent un accumulateur et un multiplieur câblés. C'est-à-dire que ces opérations élémentaires sont implantées dans le silicium et s'effectuent en un seul cycle d'horloge (lorsque les données sont immédiatement accessibles).

Les autres caractéristiques résultent de cette volonté d'obtenir une puissance de calcul optimale :

- des mémoires internes minimisant les entrées-sorties;
- une multiplication des bus de communication (interne mais aussi externe);
- une architecture pipe-line au niveau du décodage des instructions (une instruction est préparée tandis qu'une autre est exécutée). Cette technique permet d'accomplir une instruction par temps de cycle.

Une définition de plus en plus floue

Le grief que l'on pourra retenir à l'encontre de la précédente définition concerne sa trop grande généralité. Elle s'applique également à la plupart des processeurs RISC ou pseudo-CISC tels que le 68040 de Motorola.

Les DSP ne disposent pas d'un certain nombre de constituants qui demeurent l'apanage des microprocesseurs standards, à l'instar de la mémoire cache ou de l'unité de gestion mémoire. Encore faut-il relativiser ces différences par suite des exceptions qui commencent à apparaître, telles que l'annonce de l'intégration d'une mémoire cache dans la dernière version du DSP 32 bits de Motorola.

En résumé, les DSP, qui étaient destinés aux applications dédiées, se retrouvent désormais en concurrence avec des microcontrôleurs haut de gamme. En revanche, ils n'ont pas pour vocation de s'attaquer aux microprocesseurs classiques. Au chapitre des différences, les DSP ne travaillent qu'avec un seul type de données. Les microprocesseurs ont, de leur côté, une structure où plusieurs types de données se succèdent afin de répondre au mieux à leur usage généraliste. Par exemple, le couple 386SX-387SX est muni d'un bus externe sur 16 bits mais d'une structure interne sur 32 bits (entiers), tandis que le coprocesseur 387SX travaille avec des données en virgule flottante sur 32 ou 64 bits. La souplesse offerte à l'utilisateur se paye par la complexité de gestion de l'ensemble.

Un DSP est, à l'inverse, configuré pour un type défini

APPLICATIONS DES DSP

- En imagerie : reconstitution d'images, compression, simulation de systèmes optiques, restitution d'images floues par défaut de mise au point ou par bougé, compensation des turbulences atmosphériques (appliquée aux télescopes terrestres), etc.
- En audio : compression de la voix, restitution de vieux enregistrements par élimination des craquements et bruits de fond, reconnaissance vocale, synthèse de la parole, magnétophones numériques, etc.
- En télécommunication : élimination d'échos, compensation en temps réel des microperturbations sur les lignes, gestion de la correction des informations transmises par modem, téléphones portables, fax, transcodage et multiplexage dans les réseaux, gestion de protocoles télécoms, encryptage, etc.
- En informatique : accélérateur graphique, contrôleur de disques durs (avec compression des données), interpréteur PostScript pour imprimantes laser, cartes d'acquisition du son, coprocesseurs, cartes vidéo, etc.
- Dans l'armement : traitement des signaux radar et sonar, système de navigation, guidage de missiles, fusion des données issues de multiples senseurs, codage-décodage des communications hertziennes, détecteurs sismiques, etc.

Les DSP (Digital Signal Processors ou processeurs de traitement du signal) ont hérité de leur champ d'application initial, le traitement du signal, la maîtrise du calcul logique et arithmétique. Que ce soit dans les contrôleurs d'imprimantes ou dans les cartes graphiques, les DSP sortent de leur domaine de prédilection et offrent des performances à peine égalées par quelques processeurs RISC.

de données : 16 bits entiers, 24 bits entiers ou 32 bits à virgule flottante. Les bus de données (externes et internes), les registres, les mémoires, les unités de calculs sont alors tous sur le même nombre de bits. Ainsi, un DSP 32 bits flottants ne contient pas de partie dédiée au traitement des entiers qui sont considérés comme étant une sous-classe des réels et traités dans les unités flottantes. En cela, l'Intel i860 se distingue des DSP par la présence d'un cœur RISC entier sur 32 bits, en parallèle à l'unité virgule flottante qui contient des opérations câblées.

Vrais et faux DSP

Outre les processeurs RISC, la logique câblée et les processeurs en tranche sont les concurrents des DSP. La logique câblée revient à développer des circuits intégrés dédiés à l'exécution d'un algorithme donné. Si cette solution offre la puissance, elle pèche par les coûts et le temps nécessaires pour obtenir un produit fini. Elle ne donne pas non plus le droit à l'erreur : il est difficile de revenir en arrière une fois le projet lancé, comme de changer une partie de l'algorithme, ou d'effectuer une mise à jour sur le produit fini. Des chips particuliers sont cependant considérés parfois comme des processeurs de traitement du signal. C'est le cas des composants de compression d'image (C-Cube ou Intel i750).

Un processeur en tranche offre la souplesse d'adaptation, puisqu'il s'agit d'un processeur en «kit». Les constructeurs offrent un ensemble de briques de base dont l'assemblage permet de construire un processeur spécifique, chaque élément étant paramétrable. Mais les processeurs en tranche augmentent l'encombrement et le nombre de connexions (et, par conséquent, le prix de revient de la carte). La montée en puissance des DSP impose ses avantages liés à l'intégration.

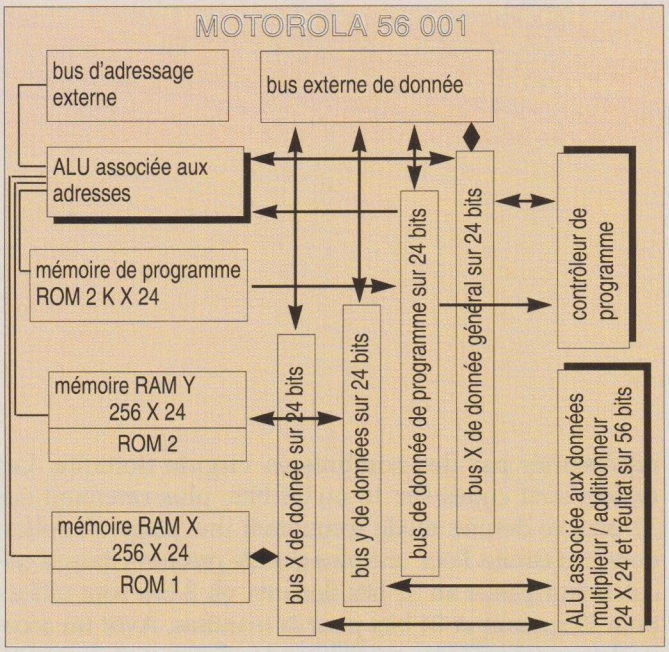
Suivant en cela les microprocesseurs, la précision de travail définit la famille à laquelle appartient un DSP. Les deux types de données traitées sont les entiers et les réels, représentés par des nombres en virgule flottante. Les entiers sont codés sur 16 ou 24 bits, plus rarement sur 32 bits, ce dernier mode recouvrant une plage d'applications très étroite. Pour une plus grande précision dans le calcul, il faut passer au 32 bits flottants, où 8 bits sont utilisés pour l'exposant et 24 bits pour la mantisse. Avec un accumulateur (chargé des additions, décalages...) et un multiplieur câblés, les DSP se sont imposés dès leur sortie en 1982, dans le traitement des données sur 16 bits entiers. La spécificité du DSP était alors claire, puisque les microprocesseurs classiques, seuls disponibles, nécessitaient près de 25 temps de cycle pour une multiplication entière sur 16 bits.

LE PRIX DE LA PERFORMANCE

La diffusion des DSP 32 bits dépend de leur prix qui définit le coût du Mflop. L'aspect économique départagera les DSP de leurs concurrents dans l'avenir. A titre indicatif, le tableau ci-dessous donne les prix constructeurs pour des quantités de mille pièces et le prix unitaire. La puissance en Mflops indiquée correspond à la valeur crête. Comme base de référence, un Intel 486 est vendu 500 dollars et offre 1 ou 2 Mflops (mais jusqu'à une précision de 80 bits).

	Pour 1 000 composants	Puissance crête en Mflops
Motorola 56001 (20 MHz)	80 $ (à l'unité) 50 $ (1000)	-
Motorola 96002(33 MHz)	500 $ (à l'unité) 320 $ (1 000)	48
Texas Instruments TMS320C30 (33 MHz)	345 $ (à l'unité)	33
Texas Instruments TMS320C40 (33 MHz)	500 $ (à l'unité)	33
AT&T DSP32C (33 MHz)	220 $ (à l'unité) 100 $ (1 000)	16

DSP CONTRE RISC



Au fil des générations, le format des données a augmenté, passant de 16 à 32 bits, tandis que la vitesse d'horloge atteint 33 voire 50 MHz. Afin de conserver la compatibilité avec les produits précédents, les microprocesseurs n'ont intégré ni unités de calcul câblées ni structure pipe-line, incompatible avec l'architecture déjà existante. Il a fallu attendre les 486 et 68040 pour qu'apparaisse l'implantation partielle de ces technologies caractéristiques du RISC. C'est la raison pour laquelle certaines opérations sont effectuées plus rapidement par un DSP que par un 68030 ou un i386, en dépit de leur complexité. Si la majorité des DSP sont des 16 bits, le Motorola 56001, à architecture 24 bits, préfigure la structure des composants haut de gamme à virgule flottante.

Un composant de transition

L'un des DSP les plus répandus est le 56001, présent sur la plupart des cartes d'acquisition et de traitement du son avec une qualité haute fidélité. Il se trouve également monté en série sur les cartes mères des stations NeXT. Il est d'ailleurs mis en avant par les logiciels de démonstration des NeXT pour ses capacités audio mais aussi graphiques.

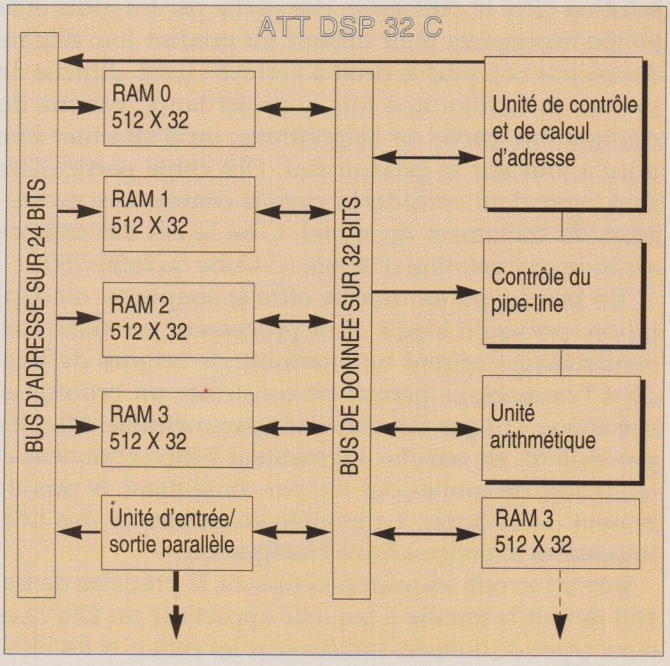
Le tracé d'une courbe fractale s'exécute ainsi plus vite avec le DSP qu'avec le processeur central (un 68040) : l'algorithme utilise des données entières et tire parti des opérations câblées du DSP (les opérateurs câblés du 68040 travaillent sur des flottants).

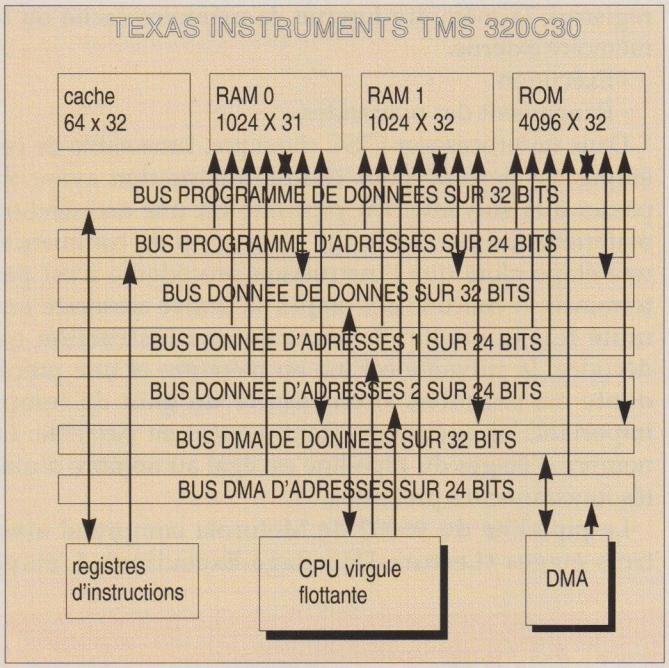
Le codage des données sur 24 bits évite l'accumulation des erreurs de calcul intermédiaire et permet d'accéder à une bonne précision sans faire appel au calcul flottant. Le 56001 se distingue par ailleurs dans les différentes formes de parallélisme interne, à savoir :

- quatre bus de données;
- trois bus d'adresse;
- une unité de multiplication et d'accumulation parallèle (UMA), avec résultat sur 52 bits;
- deux blocs de mémoires RAM appelés «X» et «Y», formés chacun de 256 mots de 24 bits, et une troisième mémoire RAM de 512 mots de 24 bits destinée aux instructions.

Chacune de ces mémoires est associée à un bus de données et un bus d'adresse, ce qui explique leur nombre. Trois unités opératoires sont actives en même temps :

- Le «Data ALU» destiné aux opérations arithmétiques et logiques et contenant les registres de travail ainsi que l'unité de multiplication-addition parallèle (UMA). Les instructions qu'il effectue sont classiques (instructions arithmétiques d'addition, de décalage, de comparaison, AND ou OR logique, test de bit, etc.). Certaines instructions comme ADDL, ADDR, SUBL et SUBR sont davantage orientées vers le traitement du signal et permettent de diviser ou de multiplier le contenu d'un registre par deux avant l'opération d'addition ou de soustraction. Le jeu d'instructions contient en plus une instruction double, MAC, qui effectue une multiplication suivie d'une addition en un temps de cycle d'instructions, et qui utilise l'UMA. A noter cependant que l'UMA n'est pas du type pipe-line (Cf.«Les coprocesseurs du 386DX et du i486» de *Langages & Systèmes* n°4).





- Le «Adress ALU». Celui-ci contient des registres. Il calcule les adresses effectives des opérandes.

- Le contrôleur de programme. Il charge une instruction, la décode puis commande son exécution. Cette unité est pourvue d'une structure pipe-line à trois étages. Le premier étage va chercher l'instruction que le deuxième décode avant que le troisième ne l'exécute.

Cette architecture parallèle à plusieurs niveaux est mise à profit par le format des instructions. Ces derniers se décomposent en quatre parties : la première indique le code de l'opération, la deuxième les opérandes sur lesquelles agira l'instruction et les deux dernières (optionnelles) qui permettent un transfert avec la mémoire interne X et un autre avec la mémoire Y.

Il résulte de tout cela une capacité de 10 Mips à 20 MHz et de 20 millions d'opérations arithmétiques par seconde. Ce composant, ainsi que les autres DSP, présente un air de famille avec les processeurs RISC, augmentés de caractéristiques issues des microcontrôleurs, à savoir la RAM interne et les capacités de liaisons multiples avec leur environnement.

La plupart des DSP étant utilisés pour traiter des données issues d'un ADC (convertisseur analogique-numérique), ils doivent faire face à certaines contraintes. Les convertisseurs délivrent en général des mots de 8, 12 ou 16 bits entiers.

Les DSP qui travaillent sur 16 bits entiers donnent théoriquement satisfaction, mais le calcul sur 24 bits offre une meilleure précision pour surmonter les erreurs de calculs intermédiaires. Il ne faut cependant pas négliger les inconvénients inhérents au calcul entier, quelle que soit sa précision.

Prenons l'exemple d'un entier non signé 16 bits. Il s'agit d'un nombre compris entre 0 et 4 095. Si l'architecture du DSP est basée sur le 24 bits, des entiers jusqu'à 16 millions peuvent être manipulés. Si l'inverse d'un nombre entier est nécessaire pour un calcul intermédiaire (par exemple, 1/4 000), la précision sur 16, 24 ou 32 bits n'apporte pas de solution. Il faut faire appel à des astuces algorithmiques, telles que la normalisation des grandeurs, pour contourner le problème, ou bien utiliser des données en virgule flottante, au prix d'une plus grande complexité interne qui engendre un coût plus élevé.

Les DSP 32 bits

L'offre des DSP 32 bits flottants est actuellement limitée aux DSP32C d'AT&T, TMS320C30 et TMS320C40 de Texas Instruments, 96002 de Motorola et ZR34325 de Zoran. Tous sont basés sur une architecture 32 bits flottants.

Ils se différencient des processeurs RISC comme l'Intel i860 ou l'AMD 29050, en dépit de la présence d'un multiplieur et d'un additionneur flottants câblés. En effet, ces processeurs RISC comportent deux unités fonctionnelles parallèles : une unité de traitement d'entiers (qui est le cœur RISC) et une unité de traitement à virgule flottante (le FPU). Le i860 utilise des instructions superscalaires, qui sont formées de deux parties, chacune destinée à une des unités.

Les DSP, en revanche, se limitent à un bloc fonctionnel unique (à l'exception du DSP d'AT&T dont l'architecture ressemble au processeur RISC avec une unité centrale et une unité arithmétique).

Leur architecture interne, du type Harvard (un bus ou un ensemble de bus est attribué aux données et un autre aux adresses), est également un élément commun : lors d'un temps de cycle, plusieurs opérations non concurrentes, comme une addition et un transfert mémoire, ont lieu simultanément.

Les spécificités de ces différents DSP résultent en partie du nombre et de la disposition des bus :

- Le Motorola 96002 prédomine dans le domaine des bus externes. Deux ports symétriques sont munis chacun d'un bus d'adressage sur 32 bits et d'un bus de données sur 32 bits et alimentent huit bus internes. Cette configuration est mise à profit dans l'architecture Maître-Esclave qui permet à deux 96002 d'être liés par l'interconnexion d'un de leurs ports respectifs.

- A l'opposé, les ingénieurs d'AT&T ont minimisé le nombre de bus internes et externes (un bus de données et un bus d'adresse), ce qui abaisse les performances, mais aussi le coût.

- le TMS320C30 emprunte la voie médiane entre les

deux cas précédents (*Cf.schémas*).

Si les DSP 32 bits présentent un grand nombre de similitudes, ils le doivent à l'implantation d'une architecture pipe-line qui est la source de leur puissance commune. Le pipe-line est également caractéristique des processeurs RISC (c'est ce qui leur permet d'effectuer une instruction par temps de cycle). Afin de comprendre le fonctionnement du pipe-line, il convient de rappeler la procédure d'exécution d'une instruction.

Cinq étapes échelonnent la progression d'une opération, quel que soit le type de processeur :

- Lecture de l'instruction («fetch»). La case mémoire contenant l'instruction, pointée par le compteur de programmes, est dupliquée dans un registre d'instruction.
- Décodage de l'instruction. Les différents signaux de commande qui déterminent l'exécution de l'instruction et l'accès aux opérandes sont générés.
- Lecture des opérandes. Elles sont situées dans les registres, la mémoire interne, la mémoire cache ou la mémoire externe.
- Exécution.
- Rangement des opérandes.

Dans un processeur CISC classique, l'ensemble de ces étapes est déroulé pour chaque instruction avant de passer à la suivante. Un pipe-line est une architecture matérielle qui permet à une instruction de commencer une étape alors que l'instruction précédente n'est pas terminée. A chacune des étapes se trouve asssociée une unité fonctionnelle. Pendant qu'une instruction est décodée, la suivante est lue en mémoire et une précédente est exécutée. Il en résulte un gain de temps important, les opérations étant menées en parallèle. Le nombre d'étages du pipe-line est égal au nombre d'unités fonctionnelles présentes.

Le pipe-line du 96002 de Motorola comprend ainsi trois étages (Lecture-Décodage-Exécution). L'étape

LES DSP DE ZORAN : DES PROCESSEURS DESTINÉS AUX TRAITEMENTS DU SIGNAL

Si la démarche générale des fabricants de DSP consiste à proposer des produits dont le domaine d'application ne cesse de s'ouvrir, d'autres ont opté pour le chemin inverse. La société américaine Zoran propose une ligne de produits exclusivement destinée au traitement du signal et à son pendant naturel, la compression d'images. Avec le ZR34325, c'est le marché des DSP 32 bits qui est visé. Concurrentielle de par la présence de trois majors de l'électronique, la stratégie de Zoran vise à éviter la comparaison directe avec les autres produits. Les deux caractéristiques essentielles du ZR34325 sont le traitement vectoriel des données et l'implantation de primitives de très haut niveau en microcode, dans la ROM interne. On trouve dans le jeu d'instructions des fonctions de traitement du signal comme la transformée de Fourier rapide (FFT) ou le filtre FIR. La mise au point est grandement facilitée et la taille du logiciel associée est fortement réduite. Comme tout DSP, le produit de Zoran comprend un multiplieur et un additionneur câblés sur 32 bits mais également un deuxième additionneur. L'architecture de bus Harvard (bus de données et d'instructions séparés) a été en revanche éliminée, le jeu d'instructions étendu permettant de réduire fortement le nombre d'instructions utilisées dans une application. De même, les voies de communication avec les mémoires externes sont limitées à un bus d'adresse sur 24 bits (16 millions de mots adressables) et à un bus de données sur 32 bits. Encore une fois, ce choix est présenté comme la résultante de l'une des caractéristiques du processeur, en l'occurrence la nature vectorielle du DSP (capacité à traiter les données par paquets ou vecteurs, et sur lesquels est appliquée la même instruction). La dénomination «vectorielle» recouvre dans ce cas uniquement le transfert par paquets entre les mémoires externe et interne ainsi que la présence dans le jeu d'instructions d'opérations agissant sur un «vecteur». Le ZR34325 traite des vecteurs de 64 mots qui sont stockés dans une mémoire RAM interne. Le gain principal provient de la minimisation du passage sur le bus. Le câblage du multiplieur et des deux additionneurs se trouve optimisé pour le cas d'un traitement de vecteurs. La puissance crête annoncée par le constructeur atteint 30 Mflops à 20 MHz (le temps de cycle instruction est égal à deux temps de cycle horloge, donc les trois blocs arithmétiques délivrent chacun un résultat tous les 100 ns). En poussant plus à fond la spécialisation, nombre de constructeurs proposent les chips de compression d'images qui constituent un cas particulier des DSP. Zoran annonce la disponibilité prochaine d'un ensemble de compression d'images, le couple ZR 36020 (processeur de transformée discrète en cosinus) et ZR 36030 (codeur-décodeur pour la compression des données issues du ZR 36020). Avec une capacité de traitement de 15 millions de pixels par seconde (codage sur 8 bits), ce composant peut s'attaquer à des applications multimédias en temps réel avec une définition vidéo de type VHS.

La solution de Zoran de compression d'images 8 bits en deux boîtiers.

«Lecture des opérandes» est comprise dans l'unité de décodage, et le rangement des opérandes est implanté dans l'étage d'exécution.

Les TMS320C30 et DSP32C ont une structure pipe-line à quatre étages. Le nombre d'étages constitue l'une des caractéristiques du pipe-line. L'autre élément à considérer est le nombre de phases nécessaire pour l'accomplissement du travail d'un étage. L'unité de temps de base d'un processeur est sa fréquence d'horloge. Par exemple, à 20 MHz, le temps de base est de 50 ns. Mais une étape de décodage ou d'exécution ne peut s'effectuer en une seule phase. Pour réaliser une étape d'instruction, il faut donc plusieurs cycles d'horloge, qui forment le cycle d'instruction.

Les DSP de Motorola et de Texas Instruments ont un cycle d'instruction composé de deux cycles d'horloge. Par exemple, pour une fréquence d'horloge de 20 MHz, le temps de cycle d'une instruction nécessaire à la traversée d'un étage est de 100 ns. Par conséquent, si le pipe-line est plein, 10 millions d'instructions seront effectuées dans ce cas. Le DSP32C a un cycle d'instruction à base de quatre cycles d'horloge. Ce composant est donc a priori deux fois plus lent que ses concurrents.

Une architecture mémoire-registre

Encore faut-il que deux phases par étage suffisent à l'accomplissement d'une tâche : la lecture des opérandes réclame plus d'un seul cycle d'instruction dans certains cas, si elles sont situées en mémoire externe. Afin d'éviter un conflit, un état d'attente est automatiquement généré, ce qui ralentit le déroulement du programme.

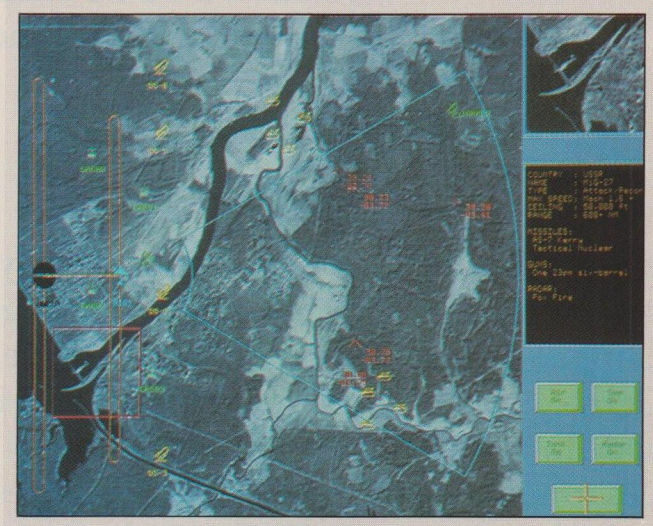
De même, si un seul cycle d'instruction suffit aux 96 002 et TMS320C30 pour l'exécution d'une multiplication et le rangement des opérandes, ce type d'opérations impose que les données soient situées dans les registres et non en mémoire. Une seconde instruction effectue les transferts registre-mémoire. Cette difficulté est cependant surmontée grâce au grand nombre de bus internes qui permettent la mise en place d'instructions parallèles. Le jeu d'instructions du DSP de Texas Instruments comprend ainsi des instructions doubles qui effectuent une opération arithmétique en même temps qu'un transfert entre les registres et la mémoire.

En conclusion, la structure en pipe-line à nombre d'étages fixes des DSP (mais également de bien des processeurs RISC) impose de répartir les instructions de manière à éviter les conflits et la perte de performance qui en résulte. Ces composants gagnent donc à être programmés avec des outils de développement de bas niveau.

DES CALCULATEURS D'IMAGES POUR LE TRAITEMENT GRAPHIQUE

Certaines applications réclament une telle puissance de calcul qu'il faut multiplier le nombre de DSP (ou de processeurs RISC), avec les problèmes de mise en œuvre d'un ensemble multiprocesseur qui en résultent, ou bien développer des circuits intégrés adaptés à l'algorithme. L'une des meilleures illustrations de ce choix est donnée par les logiciels d'analyse d'images en provenance des satellites de télédétection. Ce cas présente la particularité de demander une grande mémoire de travail et une puissance de traitement qui doit être disponible en continu. Une image stéréoscopique issue du satellite Spot peut ainsi prendre près de 200 Mo de mémoire. La principale application consommatrice de calculs est la génération d'images 3D à partir des données 2D.

Les représentations en trois dimensions interpolées à partir de mesures bidimensionnelles engendrent également d'importants moyens de calculs. L'option DSP n'est pas la plus intéressante dans ce cas puisque des algorithmes particuliers ne vont pas nécessairement bénéficier de la puissance crête des composants. D'autre part, cet exemple est basé sur la manipulation de pixels et donc n'est pas consommateur de Mflops mais d'opérations graphiques de déplacement ou d'interpolation de pixels. Il est alors rentable de développer des chips spécifiques qui remplaceront un grand nombre de processeurs programmables (DSP ou RISC). Le coût de ces composants est faible comparé à la quantité de mémoire vive indispensable. La société Theta Scan propose un produit de ce type, le VITec, destiné, entre autres, aux images satellites. Il est pourvu de quatre processeurs conçus pour cette application, accompagnés de plus de 250 Mo de mémoire vive. Présenté sous la forme d'une carte, il s'adapte sur le bus VME des principales stations de travail.



Les satellites sont de grands consommateurs de DSP.

GOLD 92

LE GRAND LOGICIEL ET

Le principe de l'«Open» est de permettre aux développeurs, lecteurs de *Langages et Systèmes*, de présenter une application professionnelle qu'ils ont réalisée en vue de l'obtention du GOLD 92 de *Langages et Systèmes*.

LES MODALITÉS DU GOLD 92

Il comporte deux pools. Chacun d'eux se verra attribuer un GOLD 92, plus deux autres prix.

1er pool : les logiciels développés au moyen d'un langage figurant dans la liste ci-dessous :

Assembleur
C (versions DOS ou Windows)
C++ (versions DOS ou Windows)
Pascal (versions DOS ou Windows)
QuickBasic
Visual Basic

2e pool : les logiciels développés au moyen de générateurs d'applications commercialisés sous la forme de package, de type Windows Toolbook, NewWave, Clipper, dBase, Foxpro, Paradox, Oracle, Magic II, Servant IV, High Screen, etc. Dans ce pool, l'utilisation d'un langage interfacé ne devra pas excéder 20% du code total de l'application développée.
Les candidats pourront — à l'aide de ces outils — présenter des applications indifféremment sous DOS, Windows et OS/2.

OPEN DÉVELOPPEMENT

RÈGLEMENT DU GOLD

Date limite de dépôt
Envoi de votre dossier d'inscription accompagné de votre logiciel à Langages et Systèmes.
Date limite : 31 août 1992.

Dossier
Votre dossier doit comporter les pièces suivantes :
1- La fiche d'inscription
Dans le cadre d'une participation en équipe, l'inscription doit se faire au nom du responsable du projet. Une fiche d'inscription sera remplie pour chaque logiciel présenté.
2- La fiche d'identité du logiciel
Une fiche d'identité doit être remplie pour tout logiciel présenté.
3- Deux exemplaires de votre logiciel sur disquette (de préférence 3,25 pouces)
4- Les sources du programme sur disquette
5- Une version de démonstration de votre logiciel
Celle-ci sera présentée sur notre serveur 36 16 INFOPC.

Pour obtenir votre dossier d'inscription, adressez votre demande à :
GOLD / Langages et Systèmes
Cedex 65 92051 Paris la Défense

Les domaines d'application concernés
Les applications développées pourront concerner les domaines suivants : gestion, scientifique, enseignement assisté, bureautique et utilitaires systèmes.

Les critères de sélection
- Intérêt pour les utilisateurs - Originalité de l'application -Convivialité de l'application - Documentation et aide en ligne - Ergonomie de l'interface - Intégrité de l'application - Exploitation des ressources matérielles.

Le jury
Le jury du GOLD sera constitué des personnes suivantes :
- Le rédacteur en chef de Langages et Systèmes
- Notre partenaire éditeur Microsoft
- Un représentant de chacun de nos partenaires du concours : Victor, Commodore et Borland.
Les décisions prises par le jury seront sans appel.

Les prix
Dans chacun des pools définis, le vainqueur se verra décerné le GOLD 92 de Langages et Systèmes.
1er prix : Victor Technologies offrira une station multimédia pour le vainqueur du premier pool. Commodore offrira un PC 486 au vainqueur du second pool. PC Soft offrira des packages de développement DOS et Windows. Microsoft et Borland offriront des packages de développement.
2e et 3eprix : Microsoft et Borland offriront des packages de développement. PC Soft offrira High Screen DOS et Windows.
La remise des prix se fera au cours du salon Software Development qui aura lieu du 15 au 17 octobre 1992.

Langages et Systèmes *se réserve tout droit d'annulation du Grand Open Logiciel et Développement et dégage toute responsabilité pour la non-remise des prix mentionnés ci-dessus.*

Enquête lecteurs

Langages & Systèmes évolue. De nouvelles rubriques sont apparues tandis que les pages de listings ont sensiblement diminué. Nous vous remercions de bien vouloir consacrer quelques minutes de votre temps pour que nous puissions mieux vous connaître et rester en phase avec vos besoins en matière de développement et de programmation.

Votre profil :

- ❒ Développeur indépendant
- ❒ Salarié d'une SSII
- ❒ Salarié d'un éditeur de logiciels
- ❒ Salarié d'un constructeur
- ❒ Membre d'une équipe de développement dans un service informatique
- ❒ Autre : _____

Vous occupez un poste de type

- ❒ Analyste programmeur
- ❒ Chef de projet
- ❒ Ingénieur système
- ❒ Autre : _____

Votre plate-forme de développement matérielle :

- ❒ Motorola 68030/040

précisez (Mac, Sun, HP, etc.) : _____

- ❒ RISC

précisez (Sun, HP, IBM, etc.) : _____

- ❒ Intel 80 286 ou compatible
- ❒ Intel i386SX ou compatible
- ❒ Intel i386DX ou compatible
- ❒ Intel i486SX/DX

Type de bus (pour les machines à base de processeur Intel ou compatible) :

❒ ISA ❒ EISA ❒ MCA

Taille de la mémoire vive : _____

Type et capacité du disque dur (SCSI, IDE, ESDI) :

_____

Configuration vidéo : _____

_____

Type et marque d'imprimante employée (laser, jet d'encre...) : _____

_____

Travaillez-vous sur une machine :

- ❒ Monoposte
- ❒ Connectée à un réseau

Dans ce dernier cas, précisez le réseau et la topologie :_____

_____

Votre plate-forme de développement logicielle :

- ❒ MS-DOS, précisez la version (5.0, 4.01, etc.) :

_____

Enquête lecteurs

❒ Windows 3

❒ OS/2, précisez la version (1.2, 1.3, 2.0) : ____

❒ Unix, précisez l'origine et la version : _____

❒ Mac OS ou System, précisez la version :

_____

❒ Autre : _____

Vos outils de développement :

❒ Vous utilisez un langage de développement. Précisez lequel et sa version : _____

❒ Vous utilisez un ou plusieurs outils de développement. Précisez lesquels et leurs versions :

_____

_____

❒ Vous utilisez un générateur d'application. Précisez lequel et sa version : _____

❒ Vous utilisez un AGL. Précisez lequel et sa version : _____

Vous développez :

❒ Des applications verticales. Précisez dans quel domaine : _____

❒ Des applications de type bureautique. Précisez dans quel domaine :_____

❒ Des utilitaires système

❒ Autres. Précisez : _____

A propos des derniers numéros de L & S

Citez les trois articles parus dans les numéros de L&S qui vous ont le plus intéressé :

_____

_____

_____

_____

Parmi les rubriques existantes, lesquelles voudriez-vous voir se développer?

❒ Re-sources
❒ Théorie
❒ Découverte
❒ Langage
❒ Unix
❒ Pixels
❒ Réseau
❒ Système
❒ Windows

Quels types de rubriques aimeriez-vous voir apparaître? _____

_____

_____

_____

_____

_____

Coordonnées facultatives

Nom : _____ Société : _____

Adresse : _____ Téléphone : _____

Merci de renvoyer ce questionnaire à Langages & Systèmes Cedex 65-92051 Paris la Défense.

info'PC

LE MAGAZINE DE LA MICRO-INFORMATIQUE PROFESSIONNELLE

Langages et Systèmes

Immeuble La Fayette 2, place des Vosges, cedex 65, 92051 Paris-la Défense 5

Tél : 49 04 79 00 - Télex : 613 234 F MONDINF

Directeur : Jacques TILLIER assisté de Catherine BOISADAM

Rédacteur en chef : François Yves LE GAL, **Assistant de la rédaction :** Serge DEL'HOMMEAU

1re secrétaire de rédaction : Catherine OLZCHANECH

Secrétaires de rédaction : Elisabeth DEROFF, Isabelle DUMONTEIL, Stéphanie GRIN

Assistant de la fabrication : Stéphane GAUTIER

Ce numéro a été réalisé sous la responsabilité de Yves Guittard

Ont également collaboré à ce numéro

Thierry BIENFAIT (rédacteur-réviseur), Marc Adler, Gilbert Chauvaux, Alexandre Coste, Alain Dorseuil, Jean-François Duvivier, Atta Oloumi, Henry-Luc Planchat, Matthieu Thfoin

La rédaction d'INFOPC dégage sa responsabilité en ce qui concerne les documents écrits, les matériels et les logiciels qui lui seraient envoyés sans demande préalable de sa part.

PUBLICITÉ

Directeur de la publicité : Sophie TALAYRACH, **Directeur adjoint de la publicité :** Nathalie TESSLER

Chefs de publicité : Virginie BRIL, Thien GILBERT, Hervé LABATTUT

Assistantes de publicité : Aromi RAJON, Laurence CARTON

DIFFUSION

Directeur : Jean PAITEL; **Responsable service abonnements :** Lucienne BOSSER assistée de Josianne LIBAN, Rose MOLNAR, Alain PACCAGNELLA, Bernadette THUAL; **Responsable des ventes :** Pierre-Olivier PROVENT

Abonnements : 49 04 79 00, 49 04 79 00; Suisse : MADITEC SA; Case postale-144CH 1020 RENENS VD1-; Tél : (021) 49 04 79 00

InfoPC est une publication éditée parMicropresse S.A. au capital de 250 000 F, en location-gérance auprès de

IDG COMMUNICATIONS FRANCE

ISSN 0981 - 6402

Commission paritaire n° 67338

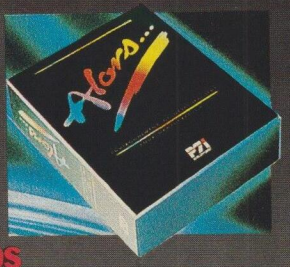
INDEX PUBLICITAIRE

BORLAND	9
COMPUSEARCH	83
COMPUTER DIRECT	82
INNOSOFT	51, 55, 65
INTERSOLV	6
ISE CEGOS	81
MICRO SIGMA	13, 41
MICROPHAR	45
MICROSOFT	3, 22
P7I DEVELOPPEMENT	81
PC SOFT	14
PC TECHNOLOGIE	4
SMARTLINE	61
TAU CETI	84

Alors... MD

UN AGL POUR NANTUCKET 5

- Dictionnaire de données
- Générateurs de sources
- Bibliothèque



La maîtrise du temps

Alors... intègre trois modules complémentaires : un dictionnaire de données, des générateurs de sources et une bibliothèque, destinés à aider le développeur à gagner du temps et créer des sources très lisibles.
A l'image des moteurs relationnels, le dictionnaire de données permet de créer les fichiers, les index, les contrôles, les liens interfichiers et d'en garantir la cohérence hors des lignes de programmes.

Un très petit nombre de lignes à maintenir

Le dictionnaire élimine toute l'écriture des contrôles interfichiers. La bibliothèque comporte une cinquantaine de fonctions majeures pour les développeurs (attribution d'un nom de fichier temporaire en réseau, log d'erreur automatique, recherche d'un champ dans un autre fichier...). Les générateurs développent un ensemble de fonctions modulaires (entrées/sorties, menus déroulants, doc. technique...).

Une maintenance aisée

Voici quelques exemples :
Un champ change de longueur? Tous les éléments utilisant ce champ fichiers, index, écrans et variables mémoires sont modifiés en cascade et reconstitués dynamiquement.
Un index est nécessaire pour un nouvel état? Vous le déclarez dans le dictionnaire et tous les programmes déjà écrits le reconnaissent immédiatement et le prennent en compte.
Pour en savoir plus renvoyez-nous le coupon ci-dessous.

PTI DEVELOPPEMENT

88 avenue Kléber - 75116 Paris
Tél. 47.27.01.29 ■ Fax 47.04.66.08

OFFRE SPECIALE 8 900 F HT 23 140 F HT 42 900 F HT
Version Monoposte ~~15 000 F HT~~ ■ 5 p. ~~39 000 F HT~~ ■ 10 p. ~~78 000 F HT~~

Je souhaiterais
- ❑ Une documentation.
- ❑ Le manuel de découverte avec la disquette de démonstration. Ci-joint un chèque de 177,90 F TTC.
- ❑ Assister à une démonstration gratuite.

Nom _____ Fonction _____
Société _____ Tél. _____
Adresse _____

Nantucket est une marque déposée par Nantucket Corporation

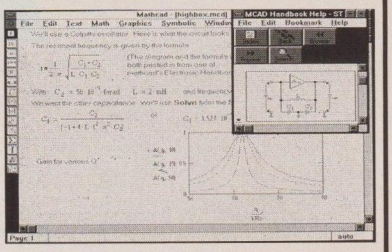
Service lecteurs n° 15 (dernière page)

Nouveau Mathcad 3.0 sous Windows

MAC PC SUN

Mathcad

Le Boss des Maths



Mathcad est un logiciel de calcul et d'édition mathématique.
Avec Mathcad, réalisez à l'écran ce que vous écrivez sur le papier : formules, équations, graphiques...et Mathcad calcule le tout pour vous !
Plus performant qu'un tableur et plus pratique qu'un langage de programmation, Mathcad est la solution des ingénieurs, techniciens, étudiants et enseignants pour les études et documents mathématiques.
Mathcad existe en environnement DOS, Windows, Mac/OS et Unix.

Fonctions principales

- Calcul numérique, logarithmique,...
- Résolution d'équations
- Symboles mathématiques
- Tracé de surface en 3D
- Import formats HPGL (AutoCad)
- Sortie PostScript
- Plus de 120 fonctions intégrées
- Modules d'applications spécifiques *(Statistiques, Electricité, Chimie,...)*

Mathcad Windows 3.0

- Calcul symbolique
- Fonctions Bessel/Fourier améliorées
- Import Bitmap
- Gestion multi-documents

Prix spécial éducation

Pour plus d'informations, appelez le (1) 46 09 24 00 ou retournez-le coupon ci-dessous

Dp. Multimédia - 204 rd pt du Pont de Sèvres
92516 Boulogne Cdx

Nom _____
Société _____
Adresse _____
Code _____ Ville _____
Tél : ( ) _____

SL 03/92

Service lecteurs n° 16 (dernière page)

Computer Direct

C'EST :

Le nouveau supplément bimestriel IDG de la

VENTE DIRECTE

105 000 exemplaires
diffusés garantis
Abonnés et ventes en kiosque d'Info PC
+ Abonnés du Monde Informatique
+ Abonnés de Télécoms & Réseaux International

200 000 acheteurs
qui le lisent (taux de circulation estimé)

0,05 Frs de coût contact

SOYEZ PRÉSENT DANS LE NUMÉRO D'AVRIL 1992:

dès maintenant, réservez vos annonces au prix le plus compétitif de la presse informatique auprès de l'équipe commerciale IDG Communications France, en appelant le

49 04 78 77

VOUS RECRUTEZ DES INFORMATICIENS

COMPU SEARCH

LE SALON DE L'EMPLOI INFORMATIQUE

LA QUALITE TOTALE

27 et 28 Mars 1992
12 et 13 Mai 1992

QUALITE DES VISITEURS

5000 informaticiens au top niveau (70 % de formation supérieure), et confirmés (79 %) ont participé au dernier Compu Search.
Motivés, ils viennent rencontrer "en direct" des entreprises, pour parler de leurs projets, de leur expérience et de leur avenir.

QUALITE DES MOYENS

Une campagne d'information percutante :
- la radio (Europe 1, Europe 2 et RFM),
- des invitations personnelles établies à partir des fichiers les plus qualifiés,
- de nombreuses annonces publicitaires et une présence dans les médias nationaux et spécialisés (Presse, TV, Radio).

QUALITE DU CONSEIL

Accueillir, conseiller les visiteurs, mais aussi les exposants, tels sont les objectifs du "Centre de Compétences Carrières". Pendant deux jours, des experts des ressources humaines dans l'informatique conseillent les visiteurs sur la gestion de leur carrière (analyse, bilan...) et les entreprises sur les méthodes d'évaluation et les politiques salariales.

QUALITE DE L'ACCUEIL

Compu Search : 40 exposants maximum afin de préserver une ambiance professionnelle et conviviale ainsi qu'un accueil personnalisé à chaque visiteur.
Chaque stand est conçu pour favoriser le contact direct et multiplier les échanges avec vos futurs collaborateurs.

LA QUALITE AU MEILLEUR PRIX

Un stand de 12 m^2, entièrement équipé, deux jours de rencontres privilégiées et ciblées avec vos futurs collaborateurs, des compétences reconnues, et une organisation fiable, pour un prix exceptionnel :
54 900 FF HT.

COMPU SEARCH

Bon à découper et à renvoyer à JL. HUGODOT - Tél. : 49 04 79 07
IDG CARRIERES - Cedex 65 - 92051 PARIS LA DEFENSE.

Société _____
Nom _____
Adresse _____
Téléphone _____

Je désire recevoir :
❒ une documentation exposant
❒ la visite d'un commercial
❒ une invitation en tant que visiteur

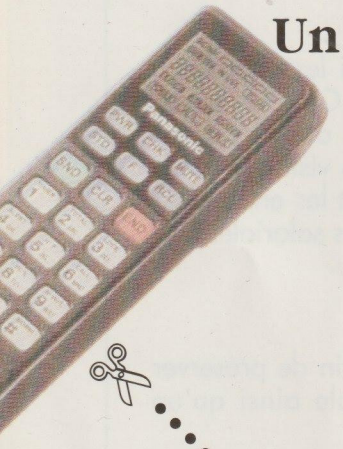
Service lecteurs n° 3 (dernière page).

TAU CETI S.A.

LES TECHNOLOGUES DE L'OBJET

SMALLTALK/V®

Un développement de 5 à 10 fois plus rapide sous OS/2 PM et Windows



SMALLTALK/V PM
SMALLTALK/V WINDOWS

PRIX UNIQUE
2 670 FRS T.T.C.
TVA 18,6 % - Franco de Port
Pour toute commande supérieure à 10 nous consulter

SMALLTALK/V est une marque déposée de Digitalk Inc, 8éme membre d'IBM International Alliance pour AD/Cycle.

OFFRE PRIVILEGE

A retourner à TAU CETI 13, rue de Lens 92000 NANTERRE

Mr Mme Melle
Société
Adresse
C.P.
Tél. :
Fonction
Ville
Fax :
OS/2PM Nbre :
Windows Nbre :
Réglement par chèque à joindre à la commande

TAU CETI S.A. - 13, rue de Lens 92000 Nanterre
Tél. : 47 80 67 61 - Fax : 47 80 29 21
S.A. au capital de 5 000 000 Frs

Toutes les marques citées sont la propriété de leurs dépositaires respectifs

Communik